IST Amigo Project
Deliverable D5.4

WP5 User Manual

IST-2004-004182
Public

Project Number	:	IST-004182
Project Title	:	Amigo
Deliverable Type	:	Report

Deliverable Number	:	D5.4
Title of Deliverable	:	WP5 User Manual
Nature of Deliverable	:	Confidential
Internal Document Number	:	amigo_d5.4_final
Contractual Delivery Date	:	7 January 2008
Actual Delivery Date	:	7 January 2008
Contributing WPs	:	WP5
Author(s)	:	**TELEFÓNICA**: José María Miranda, Álvaro Ramos
		IMS: Edwin Naroska
		TELIN: Remco Poortinga, Henk Eertink
		LogicDIS: Basilis Gladis, Otilia Kocsis
		FAGOR: Imanol Lucas, Arantxa Larrañaga
		IKERLAN: Nati Herrasti, Xavier Mardaras

Abstract

This document consists on the user manuals of the scenarios developed by the member of the AMIGO consortium implied on WP5.

Keyword list

Ambient intelligence, networked home system, home care and safety, integrated demonstrator, functional requirements, middleware, context dependent, user profiling, health centre, life cycle monitoring, entrance manager, appliance, smart washing machine, technical alarms, hardware devices, intelligent user services, integration framework

Table of Contents

Table of Contents 2

1 Introduction 8

2 User Manuals 9

2.1 Personal Health Care Centre 9

2.1.1 Provider 9

2.1.2 Development status 9

2.1.3 Intended audience 9

2.1.4 License 9

2.1.5 Language 9

2.1.6 Environment info needed if you want to run this sw (service) 9

2.1.6.1 Hardware 9

2.1.6.2 Software 9

2.1.7 Platform 10

2.1.8 Files 10

2.1.9 Documents 10

2.1.10 Tasks 10

2.1.11 Bugs 10

2.1.12 Patches 10

2.1.13 Deployment 10

2.1.13.1 System requirements 10

2.1.14 Download 11

2.1.15 Installation 11

2.1.16 Configuration 11

2.1.16.1 Application configuration 11

2.1.16.2 Network configuration 13

2.1.16.3 Compiling 13

2.1.17 Integrated applications 13

2.1.18 User Guides 14

2.1.18.1 Introduction 14

2.1.18.2 Page structure 14

2.1.18.3 Initial page 15

2.1.18.4 UPnP health devices available 16

2.1.18.5 Select health device and show actions 17

2.1.18.6 Take data 18

2.1.18.7 Show Data 19

2.1.18.8 Configure 20

2.1.18.9 Health Data 20

2.2 Daily Life Cycle Manager 22

2.2.1 Provider 22

2.2.2 Development status 22

2.2.3 Intended audience 22

2.2.4 License 22
2.2.5 Language 22
2.2.6 Environment (set-up) info needed if you wish to run this sw (service) 22
2.2.7 Platform 23
2.2.8 Files 23
2.2.9 Documents 23
2.2.10 Tasks 23
2.2.11 Bugs 23
2.2.12 Patches 23
2.2.13 Integrated Applications 23
2.2.13.1 Home Automation Gateway 24
2.2.13.2 History-Database 25
2.2.13.3 Behaviour Analyzer 26
2.2.13.4 Outlier Detector 27
2.2.13.5 Controller 28
2.2.14 User Guide 28
2.2.14.1 Habitant-System Interaction 28
2.2.14.2 Starting/Stopping the DLCM 28
2.2.14.3 Defining “Blackout Time Intervals” 28
2.2.14.4 Canceling an Alarm 29
2.2.14.5 Controlling the Warm-Up Phase 29
2.2.15 Administrator-System Interaction 29
2.2.15.1 Event Visualization 31
2.2.15.2 Event Replay File Format 32
2.2.16 Conclusions 33
2.3 Food Management 34
2.3.1 Provider 34
2.3.2 Development status 34
2.3.3 Intended audience 34
2.3.4 License 34
2.3.5 Language 34
2.3.6 Environment info needed if you want to run this sw (service) 34
2.3.6.1 Hardware 34
2.3.6.2 Software 34
2.3.7 Platform 35
2.3.8 Files 35
2.3.9 Documents 35
2.3.10 Tasks 35
2.3.11 Bugs 35
2.3.12 Patches 35
2.3.13 Deployment 35
2.3.13.1 System requirements 35
2.3.14 Download 35
2.3.15 Installation 36
2.3.16 Configuration 36
2.3.16.1 Application configuration 36

2.3.16.2 Network configuration 37
2.3.16.3 Compiling 37
2.3.17 Integrated applications 37
2.3.18 User Guides 37
2.3.18.1 Introduction 37
2.3.18.2 Tutorial 38

2.4 Health Management 42
2.4.1 Provider 42
2.4.2 Development status 42
2.4.3 Intended audience 42
2.4.4 License 42
2.4.5 Language 42
2.4.6 Environment info needed if you want to run this sw (service) 42
2.4.6.1 Hardware 42
2.4.6.2 Software 42
2.4.7 Platform 43
2.4.8 Files 43
2.4.9 Documents 43
2.4.10 Tasks 43
2.4.11 Bugs 43
2.4.12 Patches 43
2.4.13 Deployment 43
2.4.13.1 System requirements 43
2.4.14 Download 44
2.4.15 Installation 44
2.4.16 Configuration 44
2.4.16.1 Application configuration 44
2.4.16.2 Network configuration 45
2.4.16.3 Compiling 45
2.4.17 Integrated applications 45
2.4.18 User Guides 45

2.5 Appliances Management 51
2.5.1 Provider 51
2.5.2 Development status 51
2.5.3 Intended audience 51
2.5.4 License 51
2.5.5 Language 51
2.5.6 Environment info needed if you want to run this sw (service) 51
2.5.6.1 Hardware 51
2.5.6.2 Software 51
2.5.7 Platform 52
2.5.8 Files 52
2.5.9 Documents 52
2.5.10 Tasks 52
2.5.11 Bugs 52

2.5.12 Patches ... 52
2.5.13 Deployment ... 52
2.5.13.1 System requirements ... 52
2.5.14 Download ... 52
2.5.15 Installation ... 53
2.5.16 Configuration ... 53
2.5.16.1 Application configuration ... 53
2.5.16.2 Network configuration ... 54
2.5.16.3 Compiling ... 54
2.5.17 Integrated applications ... 54
2.5.18 User Guides ... 54
2.5.18.1 Actions of the Appliances Manager application on the washing machine ... 54
2.5.18.2 Actions of the Appliances Manager application on the dishwasher ... 55
2.5.18.3 Actions of the Appliances Manager application on the fridge ... 56
2.5.18.4 Actions of the Appliances Manager application on the oven ... 57
2.5.18.5 Actions of the Appliances Manager application on the boiler (central heating): ... 57
2.5.18.6 Actions of the Appliances Manager application on the induction hob ... 58
2.6 Comfort System ... 59
2.6.1 Provider ... 59
2.6.2 Development status ... 59
2.6.3 Intended audience ... 59
2.6.4 License ... 59
2.6.5 Language ... 59
2.6.6 Environment info needed if you want to run this sw (service) ... 59
2.6.6.1 Hardware ... 59
2.6.6.2 Software ... 59
2.6.7 Platform ... 60
2.6.8 Files ... 60
2.6.9 Documents ... 60
2.6.10 Tasks ... 60
2.6.11 Bugs ... 60
2.6.12 Patches ... 60
2.6.13 Deployment ... 60
2.6.13.1 System requirements ... 60
2.6.14 Download ... 60
2.6.15 Installation ... 61
2.6.16 Configuration ... 61
2.6.16.1 Application configuration ... 61
2.6.16.2 Network configuration ... 62
2.6.16.3 Compiling ... 62
2.6.17 Integrated applications ... 62
2.6.18 User Guides ... 62
2.7 Technical Alarms Management ... 64
2.7.1 Provider ... 64
2.7.2 Development status ... 64
2.7.3 Intended audience ... 64

2.7.4 License........64
2.7.5 Language........64
2.7.6 Environment info needed if you want to run this sw (service)........64
2.7.6.1 Hardware........64
2.7.6.2 Software........64
2.7.7 Platform........65
2.7.8 Files........65
2.7.9 Documents........65
2.7.10 Tasks........65
2.7.11 Bugs........65
2.7.12 Patches........65
2.7.13 Deployment........65
2.7.13.1 System requirements........65
2.7.14 Download........65
2.7.15 Installation........66
2.7.16 Configuration........66
2.7.16.1 Application configuration........66
2.7.16.2 Network configuration........66
2.7.16.3 Compiling........67
2.7.17 Integrated applications........67
2.7.18 User Guides........67

2.8 Entrance Manager........71

2.8.1 Provider........71
2.8.2 Development status........71
2.8.3 Intended audience........71
2.8.4 License........71
2.8.5 Language........71
2.8.6 Environment (set-up) info needed if you want to run this sw (service)........71
2.8.6.1 Hardware........71
2.8.6.2 Software........71
2.8.7 Platform........72
2.8.8 Files........72
2.8.9 Documents........72
2.8.10 Tasks........72
2.8.11 Bugs........72
2.8.12 Patches........72
2.8.13 Deployment........72
2.8.13.1 System requirements........72
2.8.14 Download........72
2.8.15 Installation........72
2.8.16 Configuration........72
2.8.17 Integrated applications........73
2.8.17.1 VoiceIO service integration........73
2.8.17.2 UMPS integration........73
2.8.17.3 Speaker Recognition service integration........74
2.8.17.4 Context Management service/ Face Recognition Context Source integration........74

2.8.18 User guides 74
2.8.18.1 Introduction 74
2.8.18.2 Entrance Manager as a service 74
2.8.18.3 Interaction Resources 75
2.8.18.4 User – System Interaction 76
2.8.18.5 Error Messages 77
2.9 Crisis Response Application (Anti-intrusion and Presence simulation) 78
2.9.1 Introduction 78
2.9.2 Prerequisites 78
2.9.3 CRS Component Description 79
2.9.4 Integrated Applications 80
2.9.5 User and Developer Guide 81
2.9.6 User Guide: Using CRS in the Application Mode 81
2.9.7 User Guide: Using CRS in the Service Mode 82
2.9.7.1 On Activation 82
2.9.7.2 Invoke CRS Directly 83
2.9.7.3 Invoke CRS using ANS 84
2.9.8 Developer Guide: Extending CRS for Additional ERSes 86
2.9.9 Conclusions 87
3 Conclusions 88

1 Introduction

This document summarizes the final development of the Home Care and Safety demonstrator. In the document, developers and new users of the applications will find information about the downloading process, installation and configuration and specific user guide for each application of the demonstrator. This final deliverable resumes the implementation of the requirements and specifications described in the preceding deliverables: D5.1 Home Care and Safety Functional Specification and Test Plan; D5.2 Home Care and Safety Application Specification and D5.3 Implementation of Home Care and Safety Building Blocks functionalities.

The final aim is to create a Home Care and Safety integrated prototype based on the user's requirements (specified in WP1) and using the implementation and functionalities offered by the Amigo platform (WP3) and the Amigo intelligent services (WP4). As a proof of concept of the final WP5 integrated prototype, a real home demonstrator will be use to test the applications, the user requirements, easiness, functionalities and robustness.

The entire demonstrator has been set up in Ikerlan. A new apartment has been built recently providing the minimum rooms of a little home including the kitchen, the bathroom and the living room. Also, this minimum infrastructure includes household appliances, furniture, medical measurement devices, video camera and microphone to recognize people in the entry, sensors, user location devices, displays and so on. The apartment has been designed to show the demonstrator in the most *real way* to the end-users. In this apartment the validation of WP5 applications and user's tests will be carried on to test the completeness of the user requirements.

The following picture shows the floor plan of the apartment:

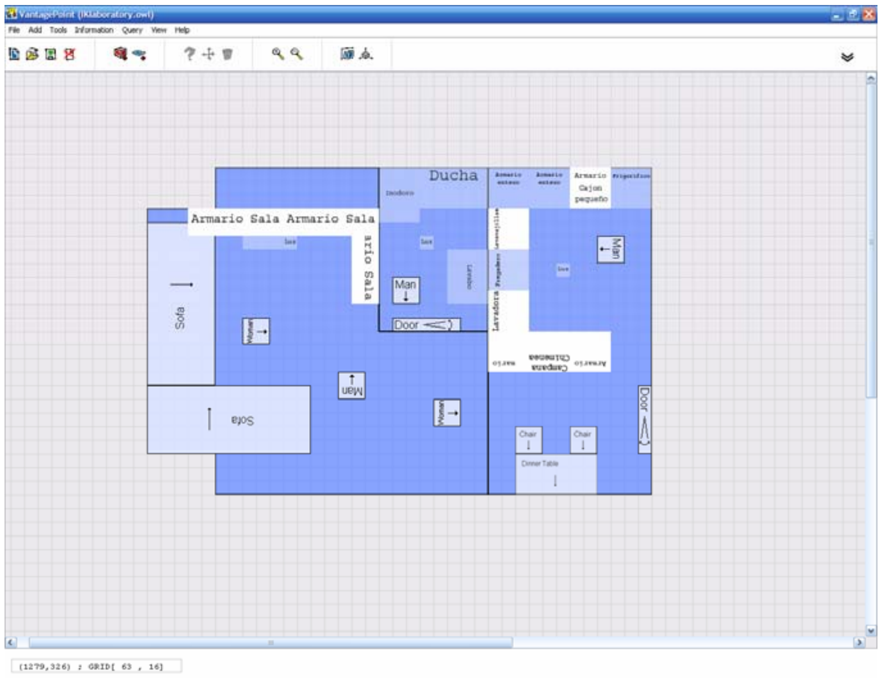

VantagePoint (IKlaboratory.owl)
File Add Tools Information Query View Help
Ducha
Armario Sala Armario Sala
Sofa
Sofa
Man
Man
Man
Door
Door
Lavadora
Campana
Chimenea
Chair
Chair
Dinner Table
(1279,326) : GRID[63 , 16]

Figure 1 Floor plan of the apartment

2 User Manuals

This document describes the user manuals of the applications developed for the " Home Care and Safety" scenario. It provides a description of the features and user relationship of the different systems that have been built as result of this WP.

2.1 Personal Health Care Centre

2.1.1 Provider

Telefónica I+D

2.1.2 Development status

2.1.3 Intended audience

Project partners who want or need to deploy and run the Personal Health Care Center application.

2.1.4 License

Proprietary.

2.1.5 Language

Java

2.1.6 Environment info needed if you want to run this sw (service)

2.1.6.1 Hardware

Minimum requirements:

- UA-767BT Blood Pressure Meter device with Bluetooth connection.
- UC-321BT Weight Scale device with Bluetooth connection.
- RFID LogiSphere infrastructure: tags of type BN-208 and HBL100-WNC receiver.
- Bluetooth USB adaptor.
- A PC/laptop with serial port, network connection and a web browser.
- A computer running Linux as BlueZ is the API used for Bluetooth communication.

2.1.6.2 Software

- Web container implementing the Servlet 2.4 and JSP 2.0 specifications
- Java Runtime Environment 1.5
- The applications that access PHC should have a web browser with HTML4.01, JavaScript Core 1.5, Cascading Style Sheets, Cookies and AJAX support.
- UPnP Health Devices web application, which is used to switch the health devices on and off.
- UPnP Context Source (to be aware of the UPnP health devices status; on or off) and its client.
- Health Data Context Source (to export all health data from a user)

- RFID server (provided by Philips) and its OSGI client.
- PostgreSQL database.
- BlueZ stack to allow Bluetooth connections.

[For environment needed by other AMIGO components needed to run MMC, please refer to the appropriate documentation]

2.1.7 Platform

Any system capable of running the software requirements.

2.1.8 Files

2.1.9 Documents

The next sections of this document provide information about deployment and configuration of the Personal Health Care Center. A tutorial is also provided in the following sections.

A general description of the Personal Health Care Center, its architecture and relation to other Amigo components are given in documents D5.1, D5.2 and D5.3.

2.1.10 Tasks

2.1.11 Bugs

None so far.

2.1.12 Patches

None so far.

2.1.13 Deployment

2.1.13.1 System requirements

The system requirements vary considerably depending on the platform chosen for the deployment and on the number of users of the application. Therefore, the following values should be evaluated by the person in charge of the system in order to adjust it properly to the actual use of the system.

A standalone server used to provide access to family members in a home environment should have the following minimum requirements:

- Pentium 1 GHz or equivalent processor
- 512 MB of RAM
- 25 MB of disk space

For the System requirements of other AMIGO components needed to run PHC, please refer to the appropriate documentation.

2.1.14 Download

2.1.15 Installation

PHC is provided as a single web application archive (war) ready to deploy on a web container. The WAR file contains the complete directory structure and all files that define the application. First of all, the web container must be stopped and any previous PHC existing directory structure should be removed.

The PHC application runs from an expanded directory structure. Web Containers or application servers vary in how the WAR file should be deployed and consequently how the expanded directory structure is created. In general, two methods are used:

1. Deploy the compressed WAR file into the web containers working directory. On some web containers (like Tomcat or IBM WebSphere), the deployment process automatically expands the WAR file into the working directory, and from that point forward, the expanded directory is considered to be the application. In this case, place the compressed WAR file in the working directory and restart the web container.
2. Expand the WAR file and deploy the expanded structure as the working directory. On other application servers (like JRun 4 and BEA WebLogic), the WAR file should be manually expanded. In this case expand the war file and place the expanded directory structure in the working directory. Finally restart the web container or application server.

The deployment method you use depends on your web container; refer to the documentation for further information on deploying a WAR file in your specific web container or application server.

The same is applicable for the UPnP Health Devices web application that is used to switch on and off the health devices.

For installation of other AMIGO components needed to run PHC, please refer to the appropriate documentation.

When installation is complete, the application can be launched at

http://hostname:port_number/phc/

2.1.16 Configuration

2.1.16.1 Application configuration

Before deploying the PHC, edit the file “ApplicationResources.properties” in the *conf* directory and set up the parameter *DataBaseURL* that should point to the IP address where the PostgreSQL database is installed. The *DataBaseUserName* and *DataBasePassword* parameters should be set to the appropriate ones. The *HealthDataCSAddress* parameter should point to the IP where it is running.

To have the context sources running, the following bundles in OSCAR are needed:

0	Active	System Bundle
1	Active	Shell Service
2	Active	Table Layout
3	Active	Shell GUI
4	Active	Shell Plugin
5	Active	Bundle Repository

6	Active	log4j
7	Active	Service Binder
8	Active	amigo_core
9	Active	amigo_ksoap_binding
10	Active	Servlet
11	Active	HTTP Service (+ Amigo mods)
12	Active	amigo_ksoap_export
13	Active	amigo_wsdiscovery
14	Active	jena-2.4
15	Active	context-broker-service
16	Active	Context Source Manager
17	Active	Context Helper
18	Active	Context Source UPnP Devices
19	Active	Context Source Client UPnP Devices
20	Active	Context Source Health Data

To have Bluetooth working correctly, the BlueZ API is needed; so in a PC running Linux install the *Build-essential* and *Libbluetooth1-dev* packages. Add in the console the following lines:

- sdptool add SP
- hciconfig piscan

The */etc/Bluetooth/pin* file should be edited and modified with the correct PIN for the health devices you are using. And finally, the ipv6 interface should be deactivated to have UPnP running correctly, so in */etc/modprobe.d/aliases* change:

Alias –net –pf –10 ipv6

by

Alias –net –pf –10 off

In the PostgreSQL database, two new tables should be created using the following scripts:

```
**Table wightscalehealthdata**

-- Table: weightscalehealthdata

-- DROP TABLE weightscalehealthdata;

CREATE TABLE weightscalehealthdata
(
  "key" numeric NOT NULL,
  userid char(30),
  weight char(10),
  date char(15),
  CONSTRAINT "pk_WeightScaleHealthData" PRIMARY KEY ("key")
)
WITHOUT OIDS;
ALTER TABLE weightscalehealthdata OWNER TO services;

**Table bloodpressurehealthdata**

-- Table: bloodpressurehealthdata

-- DROP TABLE bloodpressurehealthdata;

CREATE TABLE bloodpressurehealthdata
(
  "key" numeric NOT NULL,
  userid char(30),
  systolicpressure char(10),
  distolicpress char(10),
  pulse char(10),
  map char(10),
  date char(15),
  CONSTRAINT "pk_BloodPressureHealthData" PRIMARY KEY ("key")
)
WITHOUT OIDS;
ALTER TABLE bloodpressurehealthdata OWNER TO services;
```

2.1.16.2 Network configuration

In order to let the application work properly UPnP traffic should be allowed in the network. Note: Windows firewall does not allow UPnP traffic by default; this option should be changed in order to use the application successfully.

2.1.16.3 Compiling

Not applicable. All packages are precompiled.

2.1.17 Integrated applications

In the PHC application, other components from WP3 and WP4 are used. CMS has been widely used to get information from other applications and to provide information to other applications too. Specifically, CMS has been used to:

- Determine a user’s location: For this purpose an RFID context source has been used. The server is a Philips development, and the client subscribes to it asking for all users’ names and the rooms where they are located. Then this client makes http calls to the application’s host to provide this information to the PHC application.
- To be aware of UPnP health devices status: Whenever a UPnP health device changes its status from on to off or vice versa, a new model is created indicating the name of the device and its status, so the client just has to subscribe to it and provide this information to the application by making http calls.
- To provide health data to other applications: A user always takes a new measurement, this information is stored in a database but it is also modeled so other applications can subscribe to this context source to get this information.

UMPS has also been used to get information about users that is used in the application. For example, the photo shown when a user is present in the room is the one he has determined in UMPS.

2.1.18 User Guides

2.1.18.1 Introduction

This section thoroughly describes how the Personal Health Care application can be used in order to accomplish the needs of the user together with an in-depth description of all the problems that can arise during its usage.

2.1.18.2 Page structure

All the pages, except for the initial page, follow the same structure, allowing the user to have a fast access to the main functionalities of the application.

On the top of the page, in the left corner, the AMIGO logo will directly forward to the initial page. In the right corner, the photos of the users present in the room are shown. The users that are present in the room but not using the application appear in black and white and the picture of the user who is currently using the application appears in colour. These pictures forward to the selected user health data page. To change the user of the application, go the initial page by clicking on the AMIGO logo and select the user there.

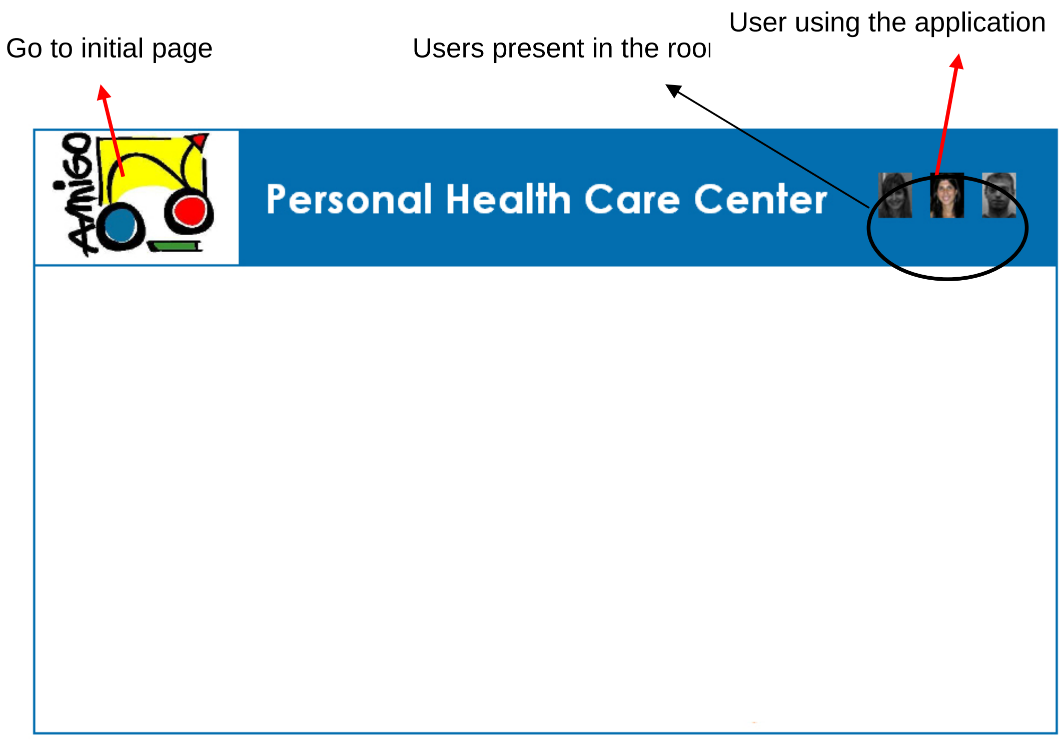


Figure 2 PHC basic structure

2.1.18.3 Initial page

The initial page of the application is where the user has to select who will be the user of the application. Only pictures of people present in the room are shown.

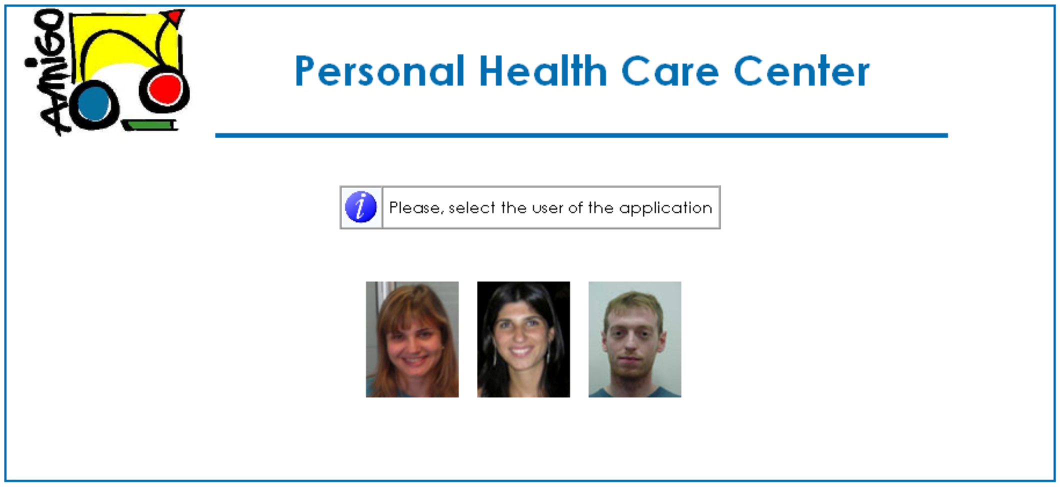

Personal Health Care Center
Please, select the user of the application

Figure 3 PHC initial page

2.1.18.4 UPnP health devices available

Once the user of the application has been set, the following page will show which UPnP health devices are available to use. If no devices have been switched on, a message will appear alerting us. The following picture shows this situation:

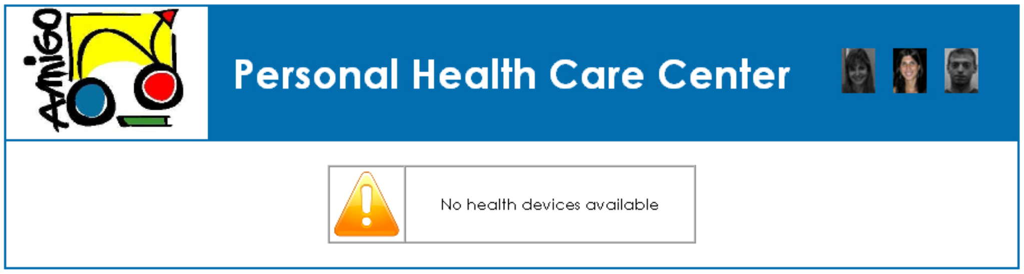


Figure 4 PHC no health devices available

To switch devices on, the UPnP Health Devices application has to be used. Devices that appear in black and white are off and they are on when they appear in colour. The following picture shows its appearance:

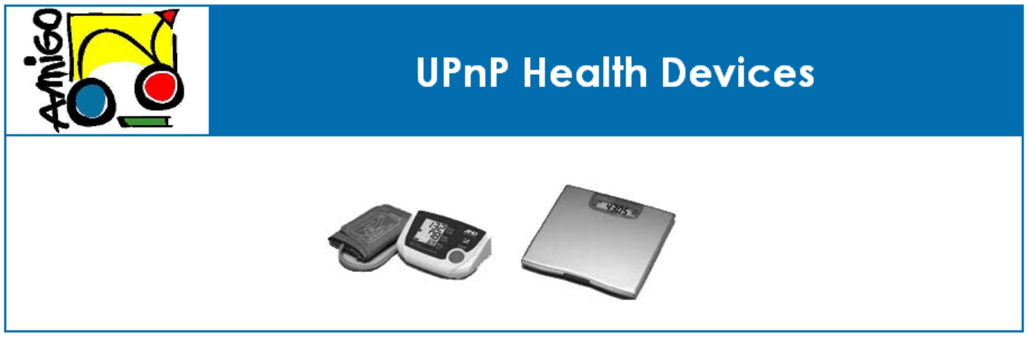


Figure 5 UPnP Health Devices, both devices off

When a device is switched on, it automatically appears in the PHC page.

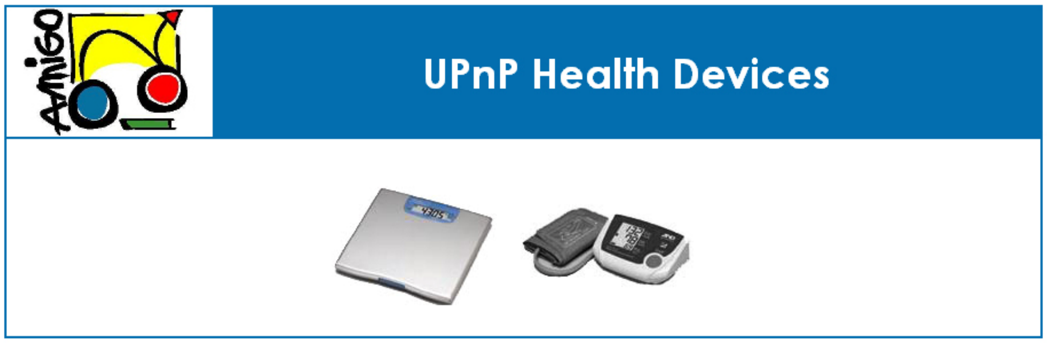


Figure 6 UPnP Health Devices, weight scale on

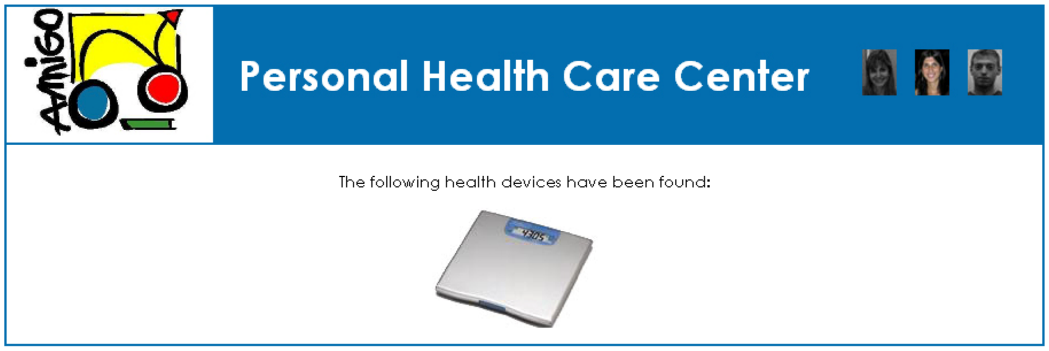

Personal Health Care Center
The following health devices have been found:

Figure 7 PHC weight scale available

When both devices are on, we will see the following picture:

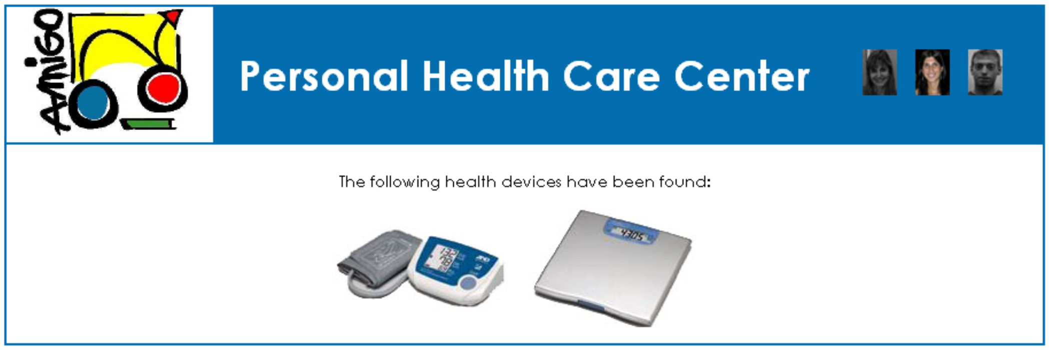

Personal Health Care Center
The following health devices have been found:

Figure 8 PHC both health devices available

Now a user can select the health device he/she wants to use.

2.1.18.5 Select health device and show actions

After selecting one of the health devices, the weight scales or the blood pressure meter, the user has to select one of the actions. The device selected will appear in the bottom left corner of the page and next to it, in the centre, the actions this device offers., The available devices will still appear at the top of the page, underneath the header, allowing the user to change the selected device whenever he wants to.

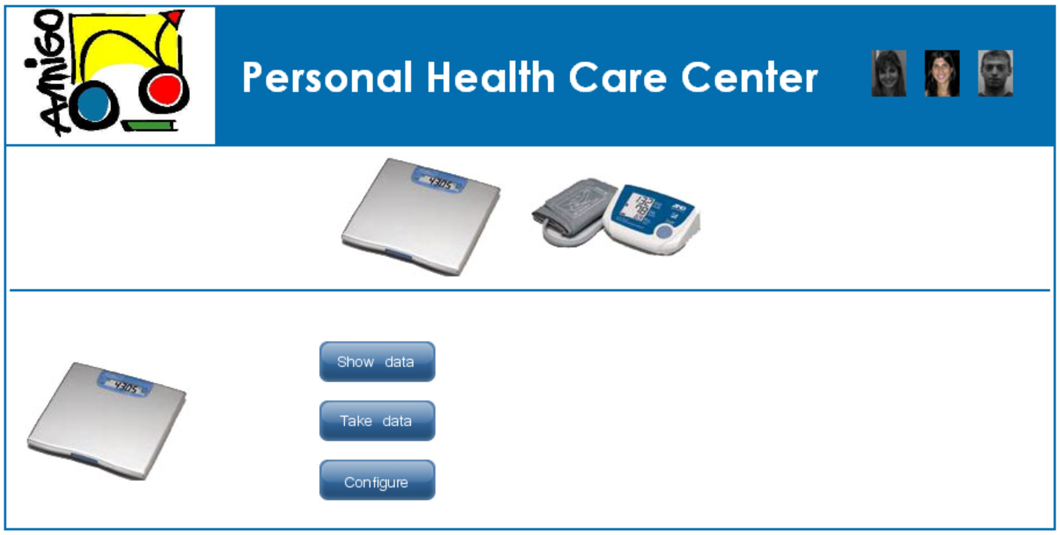
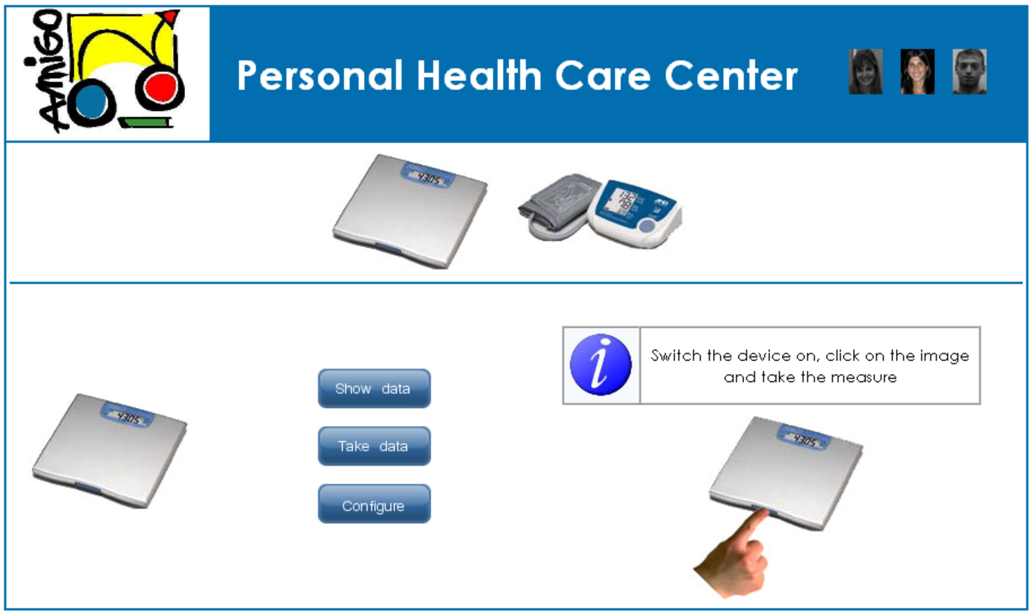


Figure 9 PHC actions

2.1.18.6 Take data

If the user wants to take a new measurement, he has to select the Take data action and then follow the instructions. The following picture shows this:



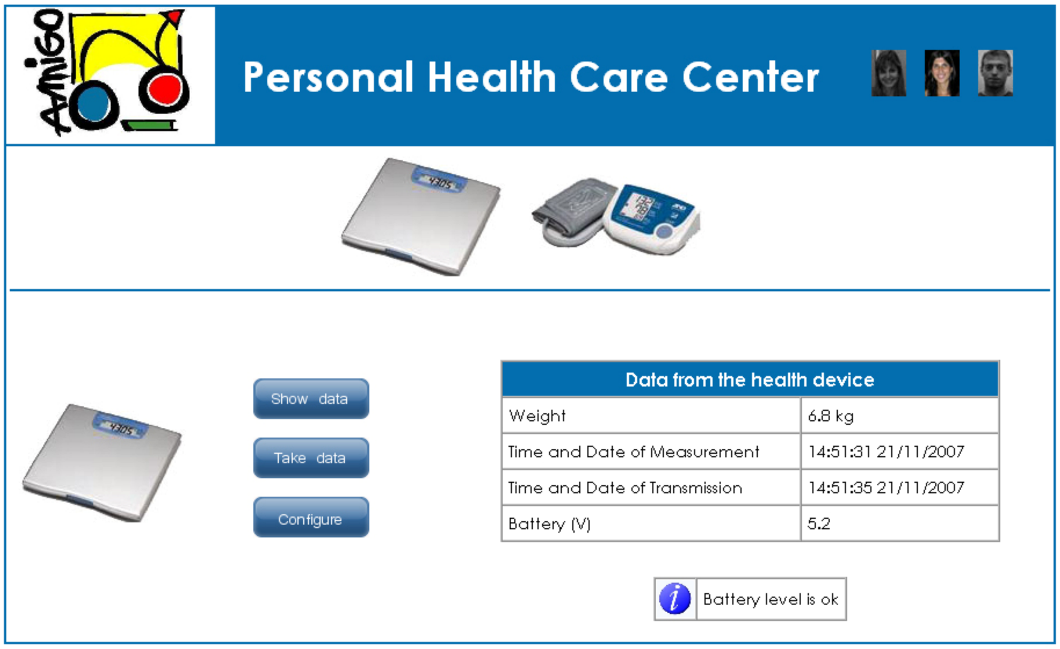
Switch the device on, click on the image and take the measure

Figure 10 PHC Take data

After taking the measurement, the PHC application will automatically show the data received.

2.1.18.7 Show Data

In this page, the user can see his last measurement and also an information message alerting about the battery status and about what their data mean in the case of blood pressure meter device.

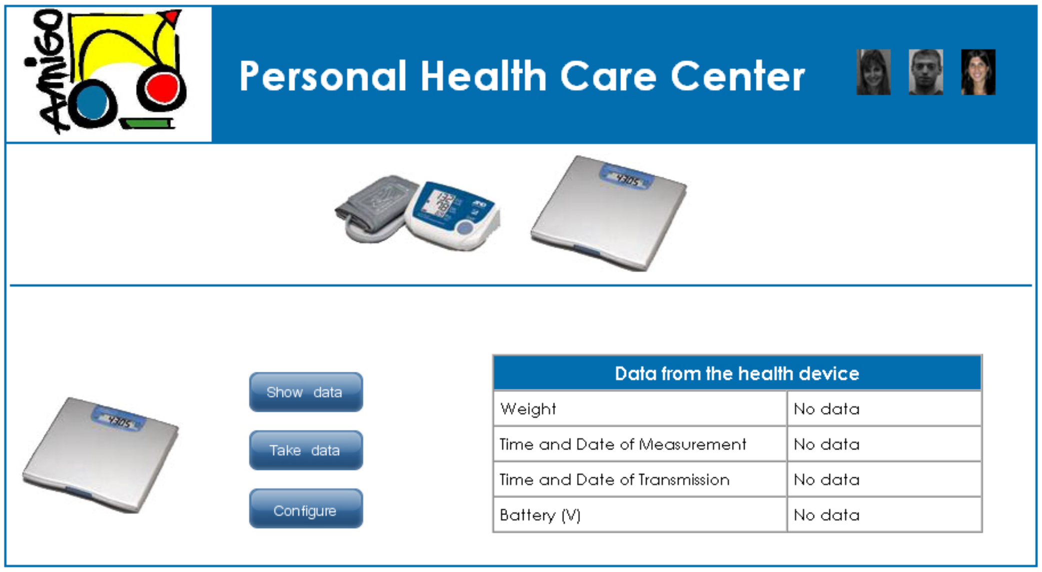


Data from the health device	
Weight	6.8 kg
Time and Date of Measurement	14:51:31 21/11/2007
Time and Date of Transmission	14:51:35 21/11/2007
Battery (V)	5.2

Battery level is ok

Figure 11 PHC Show data

If the user selects Show data action before taking a measurement, the following information will be displayed:



Data from the health device	
Weight	No data
Time and Date of Measurement	No data
Time and Date of Transmission	No data
Battery (V)	No data

Figure 12 PHC No data

2.1.18.8 Configure

This page is used to change the health device configuration. It just allows a user to change time and date information.

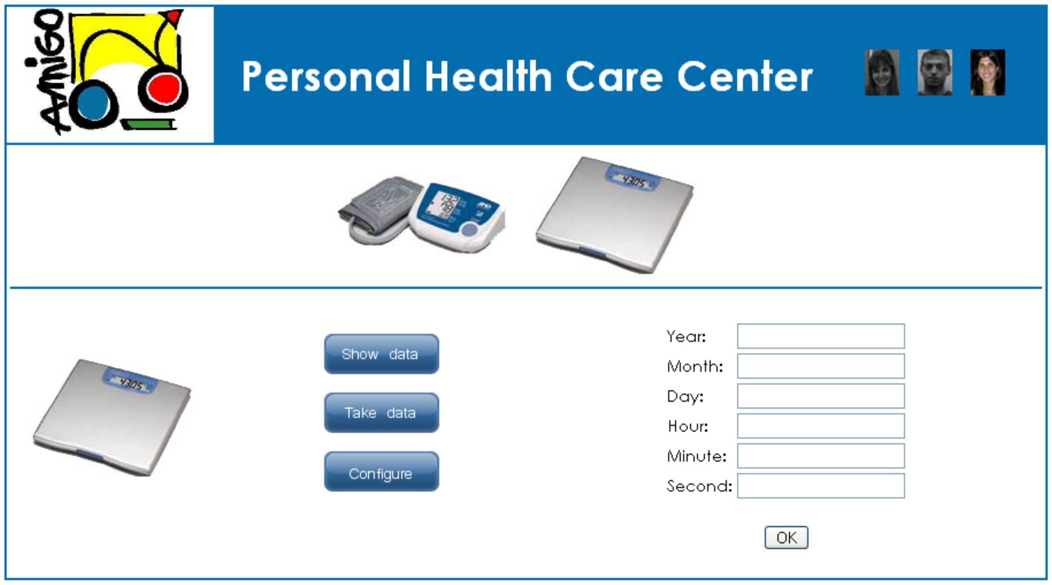
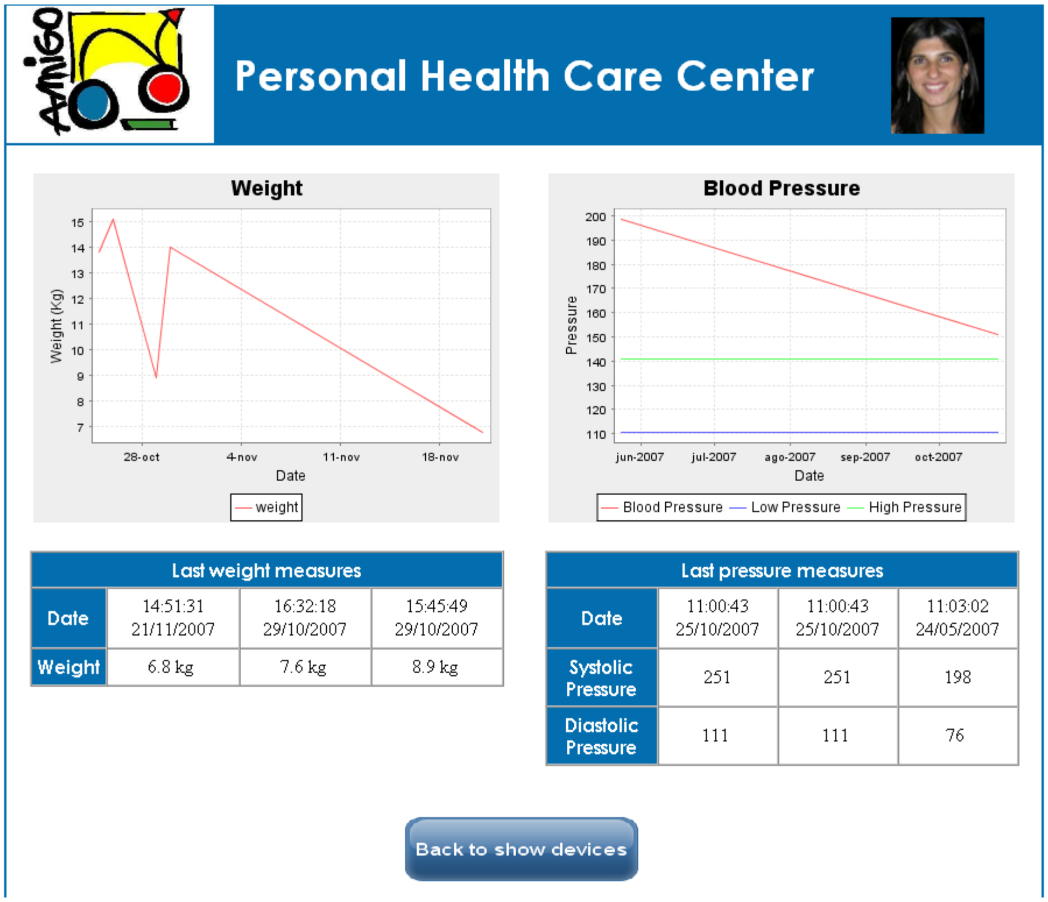


Figure 13 PHC Configure action

2.1.18.9 Health Data

Whenever a user wants to view all his health data stored in the application, he just has to click on his picture and the page shown will have the following look:



Personal Health Care Center

Last weight measures

Date	14:51:31 21/11/2007	16:32:18 29/10/2007	15:45:49 29/10/2007
Weight	6.8 kg	7.6 kg	8.9 kg

Last pressure measures

Date	11:00:43 25/10/2007	11:00:43 25/10/2007	11:03:02 24/05/2007
Systolic Pressure	251	251	198
Diastolic Pressure	111	111	76

Figure 14 PHC view user's health data

2.2 Daily Life Cycle Manager

The Daily Life Cycle Monitor (DLCM) exploits data gathered from different sensors as well as information provided by the Amigo middleware services to monitor the behavior of the inhabitant. Significant deviations from normal behavior will be detected and appropriate action will be taken. The aim of the system is to maintain the safety of the inhabitant. As a result, DLCM may be used to assist elderly people living alone at home. To this end it is able to detect if user activity is absent for a long period of time. As a result, an alarm can be raised exploiting the Crisis Response Manager.

2.2.1 Provider

Fraunhofer IMS

2.2.2 Development status

Ready.

2.2.3 Intended audience

Project partners.

2.2.4 License

The software is not public. Limited licenses are available upon request.

2.2.5 Language

Java, C#

2.2.6 Environment (set-up) info needed if you wish to run this sw (service)

Hardware: The service requires

- Windows based PC
- Typical home automation equipment connected to an EIB bus (switches, motion detectors, window sensors)
- An EIB to Ethernet bride GAMMA instabus IP interface N148/21 by SIEMENS
- PC-speakers

If no home automation equipment is available, a virtual home can be simulated playing back user activities from a log file. No special Hardware components are needed for pure log file playback.

Software: The service requires:

- .NET framework version 3.0
- OSGI framework from Prosyst version 5.2
- Amigo Middleware
- Software speech synthesis module
- The home automation gateway Alhambra 2.0 (developed by Fraunhofer IMS)
- Crisis Response Manager

- MySQL database version 5.0

Indicate all special requirements: The software makes use of the following Amigo modules / services

- Crisis Response Manager
- UPMS
- ANS
- CMS

2.2.7 Platform

JVM 5.0, .NET 3.0, OSGi 3.0

2.2.8 Files

Available on GForge with the exception of Alhambra which is available upon request (individual licensing required).

2.2.9 Documents

Design documentation.

2.2.10 Tasks

User based evaluation.

2.2.11 Bugs

No known bugs

2.2.12 Patches

none

2.2.13 Integrated Applications

The overall architecture of the DLCM application and its interplay with other Amigo components is shown in the Figure below.

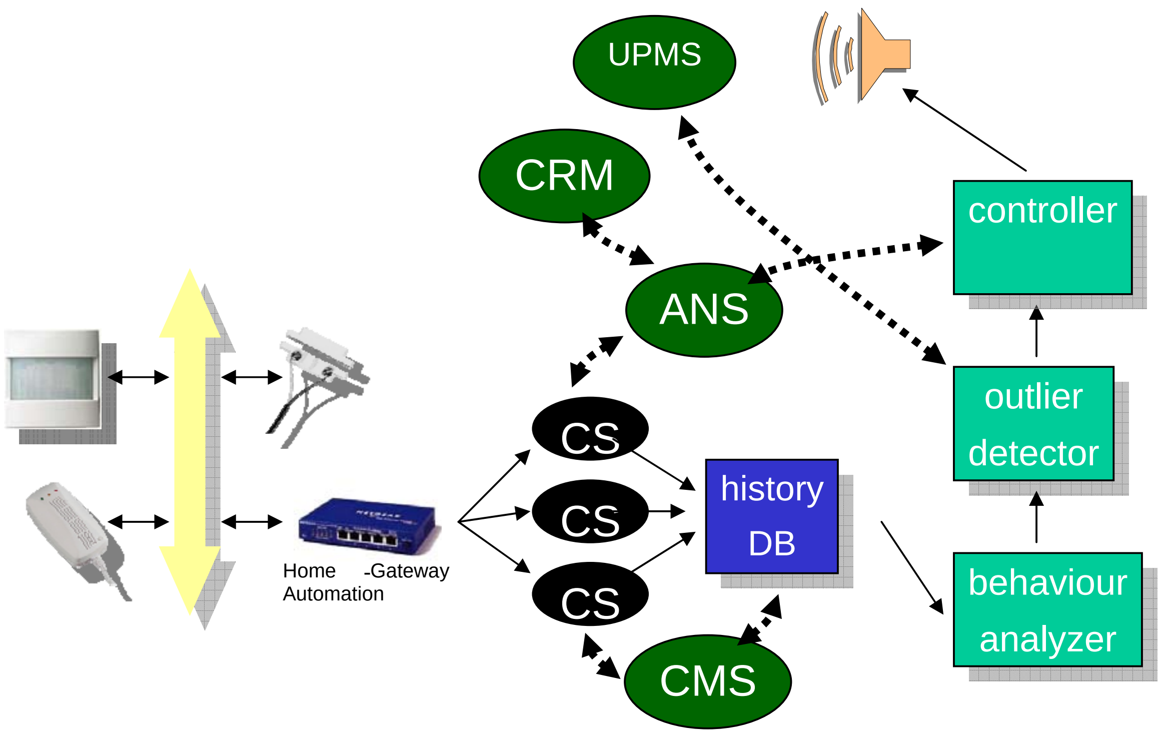


Figure 15 Architecture of the DLCM

In the following sub-sections the various components are explained.

2.2.13.1 Home Automation Gateway

The system exploits events generated by the home automation environment. Note that all kinds of events that can be electronically recorded may be considered as long as they are associated with some kind of human activity. Examples of events that can be exploited are light switches, windows sensors, and motion detectors. However, more complex "sensors" like applications that are controlled by some kind of remote control can be exploited for the DLCM as well.

In the example implementation, the sensors are connected to a home automation bus (for example, EIB or LON). In order to connect them to the Amigo world a special home automation gateway developed by the Fraunhofer IMS is used. This gateway has been extended in order to create a bridge between the home automation world and the Amigo world. Note that the gateway includes drivers for numerous home automation bus systems building a central translation instance between the low level home automation infrastructure and the high level service oriented Amigo network.

In detail, the home automation gateway generates for each home automation device an appropriate context source. This context source then represents the device in the Amigo world and sends out appropriate events each time the state of the corresponding home automation device changes. Note that this of course puts a significant burden on the overall systems as the number of events that may be generated in a large network may become significant. As a result, the number of complex events that are generated and consumed may consume a high amount of processing power.

In order to use the context sources, appropriate SPARQL query strings must be submitted by a context client. In the following the appropriate SPARQL string for the used context sources are shown.

The following text shows the SPARQL string to query a context source of type “LightStateDetectorCP” which is a binary light:

```
PREFIX context: <http://amigo.gforge.inria.fr/owl/ContextTransport.owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX light:
<http://amigo.gforge.inria.fr/owl/ims/devices/LightStateDetectorCMSModel.owl
#>
SELECT ?device_name ?state ?timestamp WHERE {
{?X context:identifier ?device_name .}
{?Y context:isSensedBy ?X .}
{?Y context:hasSensed ?Z .}
{?Z light:lightOn ?state .}
{?Y context:timestamp ?timestamp .}}
```

The next text sequence is the SPARQL string to query a context source of type “DoorStateDetectorCP” which is a door sensor:

```
PREFIX context: <http://amigo.gforge.inria.fr/owl/ContextTransport.owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX door:
<http://amigo.gforge.inria.fr/owl/ims/devices/DoorStateDetectorCMSModel.owl#
>
SELECT ?device_name ?state ?timestamp WHERE {
{?X context:identifier ?device_name .}
{?Y context:isSensedBy ?X .}
{?Y context:hasSensed ?Z .}
{?Z door:doorIsOpen ?state .}
{?Y context:timestamp ?timestamp .}}
```

The next text sequence is the SPARQL string to query a context source of type “WindowStateDetectorCP” which is a window sensor:

```
PREFIX context: <http://amigo.gforge.inria.fr/owl/ContextTransport.owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX window:
<http://amigo.gforge.inria.fr/owl/ims/devices/WindowStateDetectorCMSModel.ow
l#>
SELECT ?device_name ?state ?timestamp WHERE {
{?X context:identifier ?device_name .}
{?Y context:isSensedBy ?X .}
{?Y context:hasSensed ?Z .}
{?Z window:windowIsOpen ?state .}
{?Y context:timestamp ?timestamp .}}
```

Finally, the SPARQL string to query a context source of type “PresenceDetector” is shown. This sensor type represents a motion detector:

```
PREFIX context: <http://amigo.gforge.inria.fr/owl/ContextTransport.owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?device_name ?state ?timestamp WHERE {
{?X context:identifier ?device_name .}
{?Y context:isSensedBy ?X .}
{?Y context:hasSensed ?Z .}
{?Z context:presenceDetected ?state .}
{?Y context:timestamp ?timestamp .}}
```

2.2.13.2 History-Database

The history database searches for appropriate context sources and registers with them in order to retrieve appropriate events. As a result, it exploits the services and features provided by the Context Manager in order to locate appropriate context sources.

The history database stores the events that are received during system operation. In fact it does not only store the event type and event value but the event time stamp as well. Further, additional information about the events is gathered during operation and also stored. For example, the system analyses the events and tries to determine whether they belong to some kind of “significant” event sequence. To this end, additional data is associated with each event that support analysis.

The output of the history database is a time stamp ordered event sequence that is then used during event pattern recognition. Note that the event sequence is empty if the system is started for the first time. Hence, in order to produce meaningful output, the system must learn the behavior of the occupant for some time during a “warm-up phase”. However, during experiments it could be shown that a couple of months are sufficient as a “warming up phase”.

2.2.13.3 Behaviour Analyzer

The core of the system is the behaviour analyzer. Its basic function is to estimate the time stamp of the next user-initiated event. This estimation is then used as a reference when a new activity is expected from the user. If no activity is discovered up to this estimated time instance, an alarm is raised.

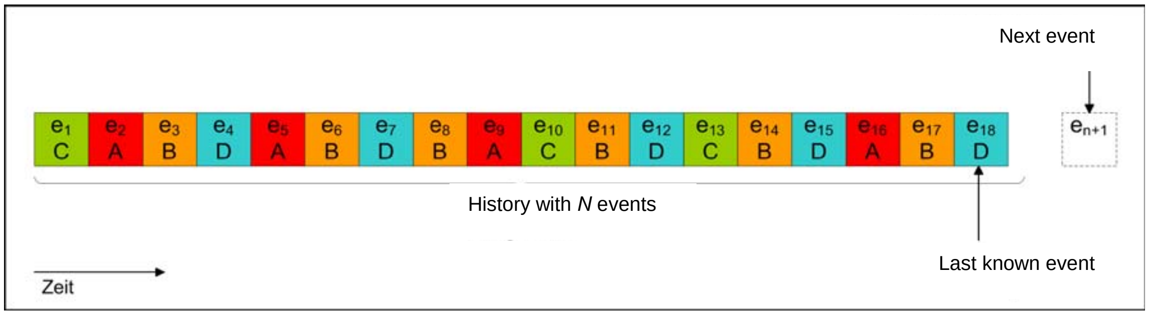


Figure 16 Event sequence list S

In the following, the procedure is described to estimate the next event time. The approach that has been used collects all the events that are recorded by the home automation environment using a database. To this end, the events are sorted in time stamp order and form the sequence list S. Let event *e* be a tuple $e_n=(a,t)$, where *a* or $a(e_n)$ are the type and value of the event (for example, the event type may be a “window sensor” and the value “open”). *T* denotes the time stamp of the event. The term *n* is a unique sequence number. The sequence list hence forms an ordered list of events $S=(e_1, e_2, e_3, \ldots e_N)$, where *N* denotes the length of the sequence and e_N is the last (with respect to its time stamp) event of the sequence list.

The next expected event time stamp is estimated as follows:

1. The sequence is compared with the same sequence shifted by *v*. In detail, *v* is running from *1* to *N – 1*. For each value of *v*, the so called hit length is calculated. The hit length is the maximum *m (m>=0)*, for which for all *j=0…m-1* the following condition holds: $a(e_{N-m})=a(e_{N-m-v})$ and $timeOfDayDifference(e_{N-m},e_{N-m-v}) < 7200$ sec. The function *a(e)* returns the type of the event *e* and $timeOfDayDifference(e_1, e_2)$ returns the time difference between two events e_1 and e_2 with respect to the day time (that is, the time difference of two events that happen at 2pm are always 0 regardless of the actual event dates). If *m>0* a hit occurred. All hits h_x for which a *m>0* holds form the so called hit set $H=\{h_a, h_b, h_b, \ldots h_x\}$. Here, the term $v(h_x)$ denotes the offset with respect to the original sequence and $m(h_x)$ the hit length of the hit h_x.

2. For all hits from the hit set *H* a score value S_h is calculated. The score value exploits the length of the longest hit sequence from *H*. This maximum length is denoted as q_{max}

in the following. The score value S_h then is $S_h = \Pi_{i=\{(v(h)\ -\ qmax\ )\ \ldots\ (v(h)\ -\ m(h))\}} P(e_i)$, where $P(e_i)$ is the frequency of the event of type $a(e_i)$ (normalized to *1*).

3. Each hit *h* is stored in the database. Along with the actual hit data additional information is stored: *Av(h)* is the average time distance between the last event from the hit and the actual event that happened. *Dev(h)* denotes the standard deviation of *Av(h)*. Both values estimate the average time to the next actual event in if *h* is in the hit set. *Diff(h)* is the time difference between the last event of *h* and the next event in the event history. Finally, the data set also stores how many times *h* was included in a hit set. This number will be denoted as *Cnt(h)* in the following.
4. In order to estimate the next event time all hits *h* from the hit set *H* are taken into account which are associated with a *Cnt(h)* value greater or equal to *1*: *H'={h in H| Cnt(h) >= 1}*. The time distance *E* to the next event is then estimated as follows: $E = C + \Sigma_{h=\{H'\}}(Av(h)+K\ Dev(h))S_h\ /\ \Sigma_{h^*=\{H'\}}S_{h^*}$. *K* is a constant parameter greater than or equal to *1*. *C* is another constant parameter greater than or equal to *0*. If the set *H'* is empty, then 98% of hits *h* from the original hit set *H* in increasing order with respect to *Diff(h)* are removed. *Diff(h)+C* from the last removed hit *h* then forms the estimated value *E*.
5. As soon as the actual event time is known (if an event finally happened), then the information for each hit *h* from the hit set *H* is updated. To this end, *Av(h)*, *Dev(h)* and *Diff(h)* are recalculated accordingly and *Cnt(h)* is incremented.

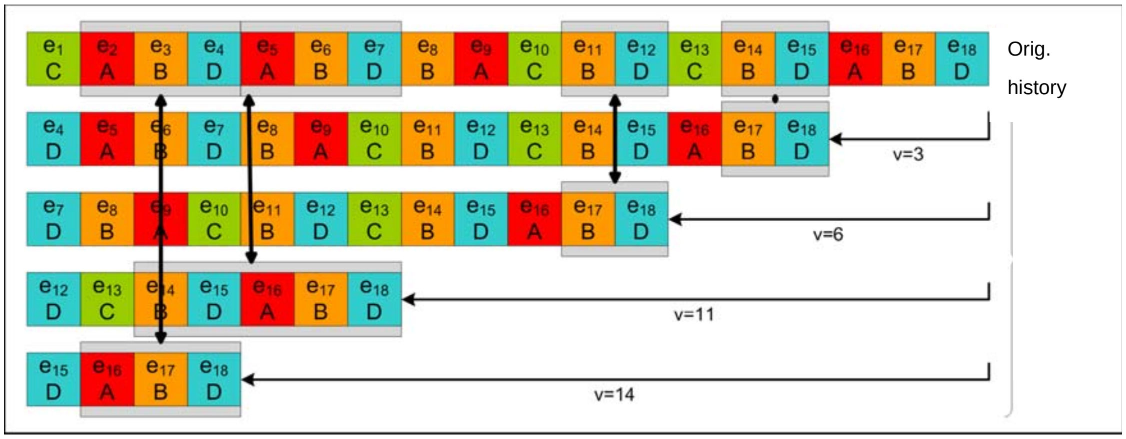


Figure 17 Search operation performed on the sequence list S

As can be seen from the description of the algorithm a variety of information is considered while estimating the next event time. For example, during pattern search only those patterns were taken into account which took place at a similar time of day. Moreover, the length of a hit is considered as well. To this end, longer sequences are favored over shorter ones.

2.2.13.4 Outlier Detector

The purpose of the outlier detector is to raise an alarm if there is a missing event. Note that the outlier detector in fact does not only take into account the information generated by the behaviour analyzer. Instead it exploits further data that is collected from the home automation sensors. For example, it detects whether an habitant (occupant) is at home or left home in order to prevent raising false alarms that occur due to the fact that nobody is at home. Note that this function is highly dependent on the actual layout of the home and the embedded sensors while the behaviour analyzer is in fact more or less independent from these parameters.

Moreover, the outlier detector also takes the typical bedtime of the habitant into account. To this end, the user can use UMPS in order to specify his or her typical bedtime intervals. During these intervals, no alarms are generated. Note that special sensors are available that sense movements or activity in the bed, so that even these “blackout times” can be avoided.

If a missing activity is detected and the alarm time is not in the “blackout interval” nor is the habitant out, an alarm signal is forwarded to the controller component.

2.2.13.5 Controller

The purpose of the controller is to give the habitant the opportunity to cancel an alarm and to manage the behaviour of the system if the alarm is finally raised. To this end, the controller uses a voice synthesis service in order ask the user for an activity that can be sensed by the home automation system (for example pressing a light switch). Note that the language of the generated voice is chosen based on the UMPS profile data. If in fact a critical situation is ongoing, the user should not do anything in order to finally acknowledge the alarm.

If the alarm is acknowledged, an appropriate context event is raised. This event is observed by ANS. An appropriate rule is installed into ANS that in turn informs the crisis response manager (CRM) if the status of the context source changes. The CRM in turn initiates all actions needed for this specific alarm type. For DLCM alarms it will ask for somebody to check the status of the habitant.

2.2.14 User Guide

As the system is not a typical user centric application, it does not provide a normal user interface. In the normal case, the DLCM stays invisible to the habitant and does not generate any messages to the user nor receives any commands directly from the user. All information that is needed by the system in order to operate properly are either collected on the fly from the home automation sensors or extracted from the user profile via UMPS.

Nevertheless, in order to show the potential benefit of the system, the application has been equipped with the option to replay the pre-recorded event sequence. To this end an appropriate administrator interface has been created.

Both, the habitant user interaction as well as the administrator user interface are explained below.

2.2.14.1 Habitant-System Interaction

As stated above, there are only a few cases where the habitant gets into contact with the DLCM: starting/stopping the service, defining the “blackout time intervals” and canceling an alarm. Finally, the user may also switch the DLCM from “warm-up” to “normal operation” status.

2.2.14.2 Starting/Stopping the DLCM

As the DLCM is running as a background service, it provides A Web-Service based interface to switch the service on or off. As a result many user interfaces can be easily used to control it.

2.2.14.3 Defining “Blackout Time Intervals”

As the DLCM observes the typical user activity and raises an alarm if no activity is sensed for some time, it needs some kind of user activity in order to operate properly. While during the typical daily life activities there are a lot of events that can be monitored by the system (motion detector events, windows that are opened or closed, light switches that are pressed, etc.) during sleep time the user typically does not provide any actions. This of course depends on the sensors that are used. For example, if bed sensors are applied that are capable of sensing motions of the user in the bed, then the system can even detect activities during sleep time. However, without the application of these special types of sensors, DLCM must be informed about the typical sleeping time of the habitant. During this time, no alarm is raised.

The information about the typical bedtime intervals (which are also called “blackout time intervals” in the following), is provided to the DLCM via UMPS. This is in fact the most convenient way to extract the information without bothering the user to provide this or similar data for perhaps several different applications repeatedly.

2.2.14.4 Canceling an Alarm

Once the DLCM detects a missing activity it raises a so called “pre-alarm”:

- During the “pre-alarm” the voice synthesis service is used to generate the following voice output “Your life cycle manager has not detected any activity from you for a very long time! Do nothing in order to raise an alarm! If you want to cancel the alarm, then press a light switch.” This provides the user with two options:
- If he or she does not press a light switch within a predefined time interval, then the alarm will be raised. To this end, the state of the context source that is represented by the DLCM will change its status. This in turn will finally activate the Crisis Response Manager to take the appropriate actions.
- If the user presses a light switch within the time interval, then the alarm is canceled and the DLCM returns to normal operation mode. Note that this special situation is taken into account to determine the future pre-alarms. As a result, the system will learn over time the specific behaviour of the user. Note that depending on the length of the warm-up phase more pre-alarms may be generated at the beginning.

2.2.14.5 Controlling the Warm-Up Phase

In order to operate properly, the DLCM needs to learn the typical behaviour of the user. To this end, it observes the event pattern generated by the home automation sensors. During the first time when no previous sensor data is available, the alarms that would be generated are very unreliable. As a result, during the “warm-up phase” no alarms are generated. Instead, the system just records and processes the events but does not generate any alarms.

Once sufficient event history information is available, the system can be switched to “normal mode”. This transition from “warm-up phase” to “normal mode” is done manually via the Web Services interface.

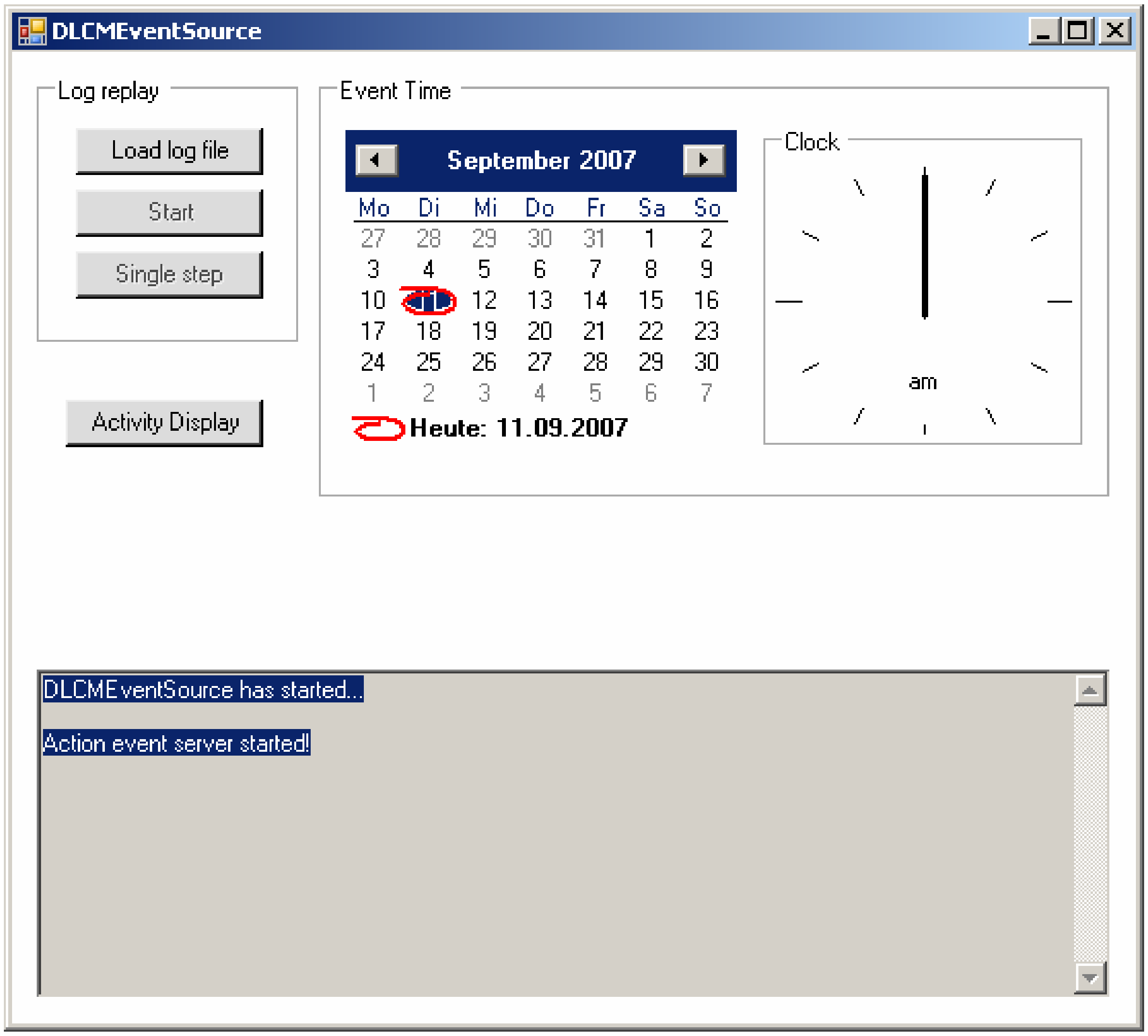
2.2.15 Administrator-System Interaction

The administrator interface can be used to observe the data generated by the home automation sensors and may be also used to re-play event sequences that were recorded in the past. The general administrator interface is shown below. It provides the following information fields:

- *Event Time*: During operation the event time and data are shown in this field. Note that this is especially useful if events are replayed as it directly shows when the events took place. During normal mode (not event replay) the time and data values will jump each time a new event is received by the DLCM.
- *Log-Information*: In the grey shaded box some log information is shown by the DLCM.
- *Log-Replay*: The log replay section allows previously recorded event sequences to be replayed. That is, the events including type of event, value of event as well as time stamp of the event are entered into the system. As a result, the DLCM will process these events as normal events. For replaying, the following actions are available:

- *Load Logfile*: If this button is pressed, a file selection box is displayed to select an event replay file. This file contains the events that are going to be entered into the system.
- *Start / Stop*: This button is used to start replaying an event sequence from the previously specified file. If replaying is started, then the events as specified by the file are replayed automatically until the file ends or the Stop button is pressed.
- *Single Step*: Each time this button is pressed, the next event from the event replay file is sent to the DLCM core. Note that this button is only available if the replay sequence is not running automatically.

• *Activity Display*: If this button is pressed, then a graphical visualization of the current event sequence is shown.



```
DLCMEventSource has started...

Action event server started!
```

Figure 18 Administrator control interface

2.2.15.1 Event Visualization

In order to easily follow the behaviour of the habitant and also to debug and reason about the behavior of the DLCM a graphical visualization component has been incorporated into the DLCM. This component can take a vector graphic in SVG format and animate it according to the events that are sent to the DLCM.

SVG is a XML based vector graphics format. To show SVG graphics, a freely available SVG render program has been integrated into DLCM and modified appropriately. To add a sensor to the graphics the following steps must be done:

1. The room layout graphics must be loaded into a SVG editor program. Note that one may use an “image to vector graphics converter” in order to generate appropriate SVG layout graphics from a scanned layout plot.
2. The sensor as it shall be shown if it is activated is drawn with the help of the SVG editor. The sensor graphics element is then grouped (if necessary) and an “ID” tag is added to the element. The value of the ID tag shall get the same name as specified by the corresponding context source that represents the sensor.
3. Save the file as “flat5.svg” in directory “DLCM/bin” sub-directory.

An example activity display window is shown in the Figure below:

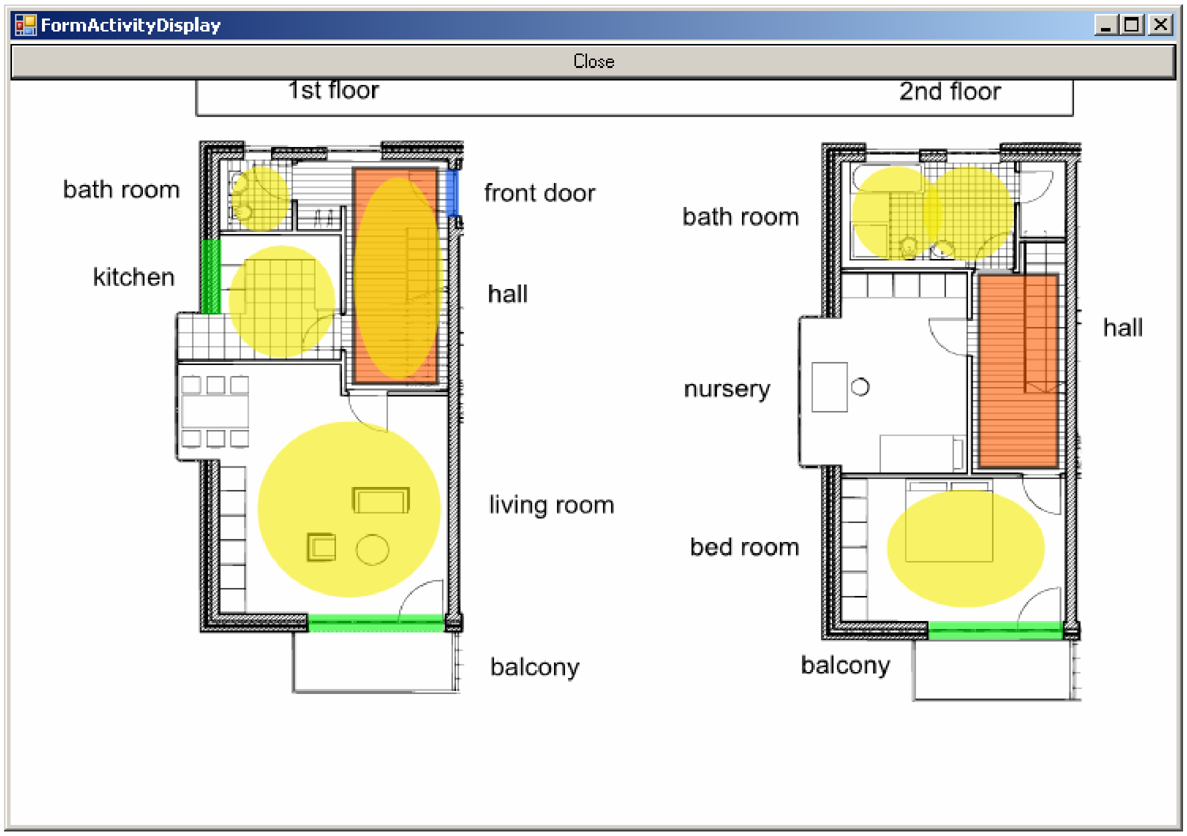


Figure 19 Sample activity window

In this example, an apartment that extends over two floors is shown. Note that all available sensors are shown in the figure. Depending on the sensor type, they are plotted as yellow circles (indicating lights), red boxed (indicating motion detectors), green bars (indicating window sensors), or blue bars (indicating a door sensor). During operation the graphical elements are displayed or disappear according to the state of the corresponding sensor.

2.2.15.2 Event Replay File Format

In the following the file format for the replay file is explained.

The actual file contains of two parts. In the first part (header part) a list of all available events are provided. The second part is the actual list of events.

2.2.15.2.1 Available Event Types

An example sequence for the definition of available events is given below:

```
FlurUnten-BE1 user -> ' F T ' [GROUP:0/0/1]
Schlafen-BE1 user -> ' F T ' [GROUP:0/0/3]
FlurUnten-TK1 user -> ' F T ' [GROUP:0/0/2]
Wohnen-BE1 user -> ' F T ' [GROUP:0/0/4]
Schlafen-FK1 user -> ' F T ' [GROUP:0/0/5]
Kueche-BE1 user -> ' F T ' [GROUP:0/0/6]
Kueche-FK2 user -> ' F T ' [GROUP:0/0/7]
Bad-BE1 user -> ' F T ' [GROUP:0/0/8]
FlurUnten-BW1 system -> ' F T ' [GROUP:0/0/9]
Abstell-WC-BE1 user -> ' F T ' [GROUP:0/0/10]
FlurOben-BW1 system -> ' F T ' [GROUP:0/0/11]
Bad-BE2 user -> ' F T ' [GROUP:0/0/12]
```

Each line defines a kind of event. This first token is the event name, followed by the keyword “user” defining that the event is caused by the user or “system” if the event is generated by the home automation system (for example an event that indicates that we are currently at the weekend is a system generated event). After the token “->” the possible values of the event are listed between ‘ ‘. In case of on Boolean event F for false and T for true is printed. Finally, for EIB components the header part defines the EIB group address in “[]” brackets for each corresponding EIB component.

2.2.15.2.2 Event Sequence

An example sequence is shown below. Note that the sequence starts with the token “simulation:”. Note further, that each line is a separate event. In detail, each line defines the new state of the system after the event took place.

```
simulation:
time = 1148461633  weekend = F  FlurUnten-BE1 = F  FlurUnten-TK1 = F
Schlafen-BE1 = F  Wohnen-BE1 = F  Schlafen-FK1 = F  Kueche-BE1 = F  Kueche-
FK2 = F  Bad-BE1 = F  FlurUnten-BW1 = F  Abstell-WC-BE1 = F  FlurOben-BW1 =
T  Bad-BE2 = F  #
time = 1148461649  weekend = F  FlurUnten-BE1 = F  FlurUnten-TK1 = F
Schlafen-BE1 = F  Wohnen-BE1 = F  Schlafen-FK1 = F  Kueche-BE1 = F  Kueche-
FK2 = F  Bad-BE1 = F  FlurUnten-BW1 = F  Abstell-WC-BE1 = F  FlurOben-BW1 =
F  Bad-BE2 = F  #
time = 1148461655  weekend = F  FlurUnten-BE1 = F  FlurUnten-TK1 = F
Schlafen-BE1 = F  Wohnen-BE1 = F  Schlafen-FK1 = F  Kueche-BE1 = F  Kueche-
FK2 = F  Bad-BE1 = F  FlurUnten-BW1 = F  Abstell-WC-BE1 = F  FlurOben-BW1 =
T  Bad-BE2 = F  #
time = 1148461663  weekend = F  FlurUnten-BE1 = F  FlurUnten-TK1 = F
Schlafen-BE1 = F  Wohnen-BE1 = F  Schlafen-FK1 = F  Kueche-BE1 = F  Kueche-
FK2 = F  Bad-BE1 = F  FlurUnten-BW1 = F  Abstell-WC-BE1 = F  FlurOben-BW1 =
F  Bad-BE2 = F  #
time = 1148461684  weekend = F  FlurUnten-BE1 = F  FlurUnten-TK1 = F
Schlafen-BE1 = F  Wohnen-BE1 = F  Schlafen-FK1 = F  Kueche-BE1 = F  Kueche-
```

```
FK2 = F  Bad-BE1 = F  FlurUnten-BW1 = F  Abstell-WC-BE1 = F  FlurOben-BW1 =
T  Bad-BE2 = F  #
time = 1148461692  weekend = F  FlurUnten-BE1 = F  FlurUnten-TK1 = F
Schlafen-BE1 = F  Wohnen-BE1 = F  Schlafen-FK1 = F  Kueche-BE1 = F  Kueche-
FK2 = F  Bad-BE1 = F  FlurUnten-BW1 = F  Abstell-WC-BE1 = F  FlurOben-BW1 =
F  Bad-BE2 = F  #
```

Each line defines the complete state vector of the home automation system providing the value for each of the sensors. The first token in each line is “time = “ specifying the time tic of the event. Note that the time tic is the UNIX time tic (seconds elapsed since 1970). It can be easily converted back into local time values. Note further, that the state or the sensors are provided in the format “sensor-name = sensor-value”.

2.2.16 Conclusions

The Daily Life Cycle Manager is an attractive application that shows the power and benefits of a smart home automation system compared to the current typical home automation systems. The DLCM exploits data gathered from different sensors as well as information provided by the Amigo middleware services to monitor the behaviour of the inhabitant. Significant deviations from normal behaviour will be detected and appropriate action will be taken. The aim of the system is to maintain the safety of the inhabitant. As a result, DLCM may be used to assist elderly people living at home alone. The system is capable of detecting if user activity is absent for a long period of time. As a result, an alarm can be raised exploiting the Crisis Response Manager

Using Amigo middleware for the DLCM application provides several advantages:

- Due to the abstraction of context sources, various kinds of information may be exploited by the DLCM in order to monitor the behaviour of the habitant. As a result, the actual monitor and pattern recognition algorithms can operate at a higher level of sensor data. As a result, they do not need to deal with the actual basic sensor details like the bus system that is used to connect them to the network.
- The DLCM application can make use of many tools and services that are provided by the amigo middleware. For example there is no need to implement complex lookup and discovery mechanisms into DLCM in order to find all appropriate context sources. Instead, the functionality of the Context Manager can be used.
- User specific data can be easily extracted from UMPS in order to tailor the DLCM application towards the specific user. UMPS allows accessing this data in a standardized and consistent way without the need to implement application specific user interfaces.
- Common services that are needed by several applications can be easily “outsourced” in order to make the system more consistent. For example, the Crisis Response Manager has been identified as a component that is needed by several applications to deal with situations that require special emergency handling.

Of course, there is also a price to pay when using this kind of middleware: The home automation sensors generate numerous events that are forwarded to the appropriate Amigo proxies and converted into corresponding “Amigo events”. As a result, numerous SPARQL queries are generated and must be processed. This in turn may create a significant processing overhead which of course has a bad impact on the total cost of ownership of such a system.

2.3 Food Management

2.3.1 Provider

Ikerlan

2.3.2 Development status

Functional prototype

2.3.3 Intended audience

Project partners who want or need to deploy and run the Food Management application.

2.3.4 License

Proprietary.

2.3.5 Language

.Net C#

2.3.6 Environment info needed if you want to run this sw (service)

2.3.6.1 Hardware

Minimum requirements:

- Fridge with RFID infrastructure (OBID RFID 13,56 Mhz reader) and communications resources through Power Line.
- Wireless Weight Scale Medic4all with RF communication.
- RFID infrastructure: 13,56 Mhz ISO tags applied to different foods.
- PDA (shopping list application) with WiFi.
- A PC with communication resources for Power Line, WiFi and RF.

2.3.6.2 Software

- Windows PC Software
- .Net Framework 2.0
- AMIGO Middleware
- (Optional) Home Automation Sensor Information (get fridge content)
- OWLDotNetApi (OWL parser)
- NxBRE libraries (rule engine)
- Flash.ocx (integrate Flash Interfaces on Applications)
- Java Runtime Environment 1.5
- Medical4all software for health measure acquisition

- MD – Gateway – Helper to create all Context Sources
- OBID RFID driver
- Food Management Application

2.3.7 Platform

Any system capable of running the software requirements (.Net, Windows).

2.3.8 Files

Available in GForce

2.3.9 Documents

Next sections of this document provide information about the Food Management Application. Also a tutorial is provided in the next sections.

General description of the Food Management Application, its architecture and relation to other Amigo components are given in documents D5.1, D5.2 and D5.3.

2.3.10 Tasks

Integration with other components from AMIGO system to develop a fully functional application as described in document D5.2.

2.3.11 Bugs

None so far.

2.3.12 Patches

None so far.

2.3.13 Deployment

2.3.13.1 System requirements

System requirements exclusively depend on the features of the application. Thus, a typical platform with the following minimum requirements will be needed:

- Pentium 1 GHz or equivalent processor
- 512 MB of RAM
- 25 MB of disk space
- Minimal Screen resolution must be 102x768 (full screen)

For System requirements of other AMIGO components needed to run Food Management Application, please refer to the appropriate documentation.

2.3.14 Download

Files required to run the Food Management Application can be downloaded from the Inria GForce repository available at:

/svn+ssh://login@scm.gforce.inria.fr/svn/amigo

2.3.15 Installation

The Food Management Application is provided as a software application ready to deploy on a .NET platform. A specific application is being developed to install the Food Management Application in the AMIGO environment.

2.3.16 Configuration

2.3.16.1 Application configuration

The following AMIGO bundles in OSCAR are needed:

0	Active	System Bundle
1	Active	Shell Service
2	Active	Table Layout
3	Active	Shell GUI
4	Active	Shell Plugin
5	Active	Bundle Repository
6	Active	log4j
7	Active	Service Binder
8	Active	amigo_core
9	Active	amigo_ksoap_binding
10	Active	Servlet
11	Active	HTTP Service (+ Amigo mods)
12	Active	amigo_ksoap_export
13	Active	amigo_wsdiscovery
14	Active	jena-2.4
15	Active	context-broker-service
16	Active	Context Source Manager
17	Active	Context Helper

The application loads the configuration variables from the CookingAssistant.exe.config file, such as user profile, output directory for the week planning, etc.

To have RF communications working correctly, the RF driver provided by Medic4all have to be installed.

To add new products with new RFID codes an XML file must be changed. This XML (Products.xml) is located under the Config Directory in the CMS component called Gateway (Developed by Ikerlan in WP4) .

In the cmconfig.xml file the port that supports the RFID communication has to be specified.

2.3.16.2 Network configuration

In order to let the application work properly, it is best that all the modules are installed in the same platform.

2.3.16.3 Compiling

Not applicable. All packages are precompiled.

2.3.17 Integrated applications

From WP3, AMIGO Middleware based on .Net has been used.

From WP4: Context Sources from CMS (fridge); UMPS (user's preferences); UIS (visual).

2.3.18 User Guides

2.3.18.1 Introduction

The Food Management Application (FMA) deals with:

- Food at home
- The management of the goods in the fridge
- The use-by date of the goods
- A set of recipes

It also deals with the profile of each family member related to food, ingredients liked, those disliked, possible diseases like cholesterol, diabetes and so on. Each family member sets his/her preferences and his/her height, weight and gender. Also, each family member specifies when she/he is going to have breakfast, lunch or dinner the following week and, based on the info provided, Food Management determines which type of people is going to eat the same menu.

Performing reasoning analysis on data related to each meal, the FMA suggests one suitable menu for the whole family for each meal and arranges the menu planning for the next week, generating a shopping list by checking the ingredients available in the fridge. If the FMA detects a product that needs to be used soon, the menu will be changed to include a recipe that uses that ingredient.

All the information related to ingredients, food type, characteristics, recipes, diseases, user's body variables and so on is structured with ontologies. The reasoning process combining user profiles, preferences, ingredients, recipes, user's agenda, diseases, special diets, etc. is performed by means of rules written in RuleML and managed by a rule engine in .NET.

The following paragraphs describe the functionality of the FMA and the relationship between this application and the users.

Figure 20 FM initial page

2.3.18.2 Tutorial

The main menu will let you move between different screens of the application. This menu is located on the upper side of the screen. The fist task to be done is to enter the users and their profiles into the system. Each profile can be edited or deleted.

In the user profile, the user must indicate the ingredients that the user likes, dislikes or recommends, any diet that the user could be following or any medical advice.

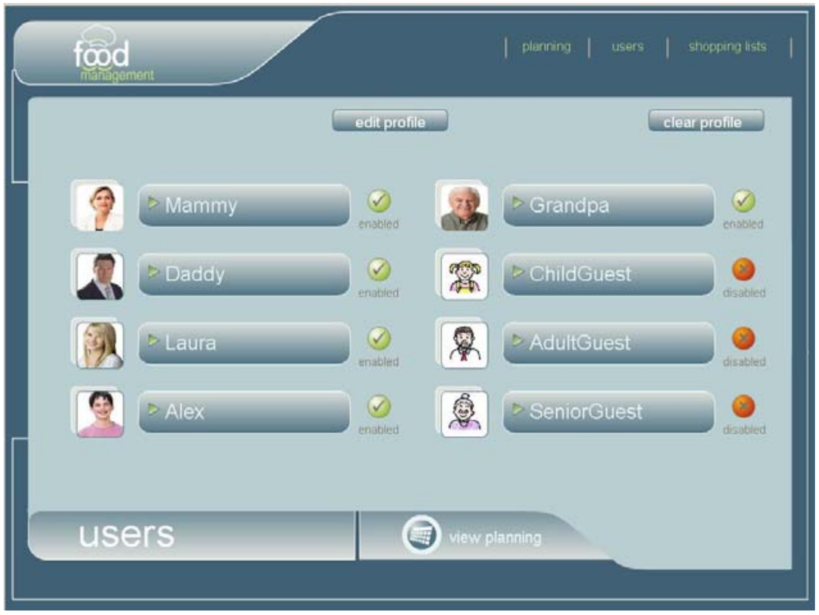
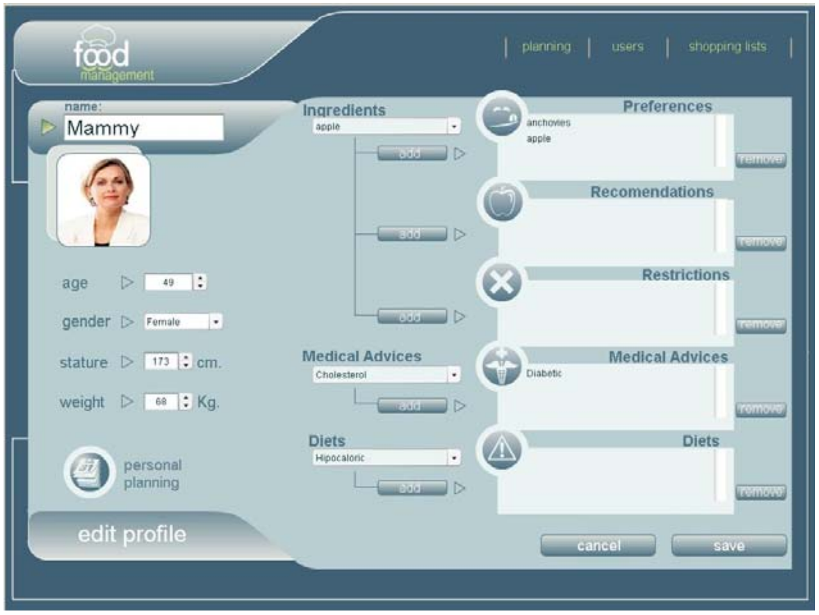


Figure 21 FM users list



name: Mammy

age	49
gender	Female
stature	173 cm.
weight	68 Kg.

Ingredients: apple

Medical Advices: Cholesterol

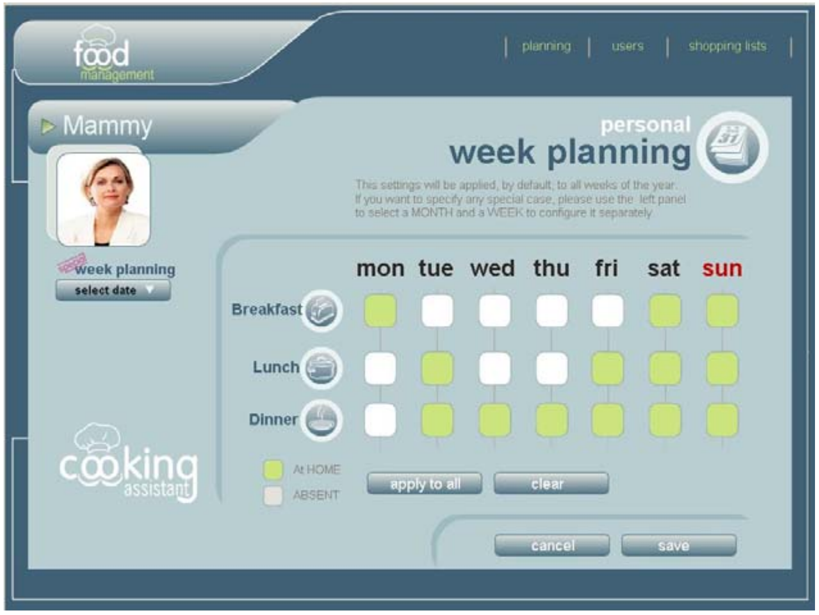
Diets: Hipocaloric

Preferences: anchovies, apple

Medical Advices: Diabetic

Figure 22 FM user profilet

Furthermore, each user must add the user week planning (the days of the week that the user will be at home to have breakfast, lunch or dinner).



This settings will be applied, by default, to all weeks of the year.
If you want to specify any special case, please use the left panel to select a MONTH and a WEEK to configure it separately

	mon	tue	wed	thu	fri	sat	sun
Breakfast	🟩	⬜	⬜	⬜	⬜	🟩	🟩
Lunch	⬜	🟩	⬜	⬜	🟩	🟩	🟩
Dinner	⬜	🟩	🟩	🟩	🟩	🟩	🟩

🟩 At HOME
⬜ ABSENT

Figure 23 FM user calendar

Once the user inserts all the profiles, the User Diets Advisor application will evaluate the conditions and a new suggestion will be generated for the whole week.

These user profiles are stored automatically in a XML file, which will be loaded when the application is initializing.

Ingredients, diets and medical advice are loaded from the recipe ontology.

The suggestion can be consulted on the planning screen. The system will generate a suggestion for each day of the week and any lunch time of the day when any user will be at home according to each user's week planning.

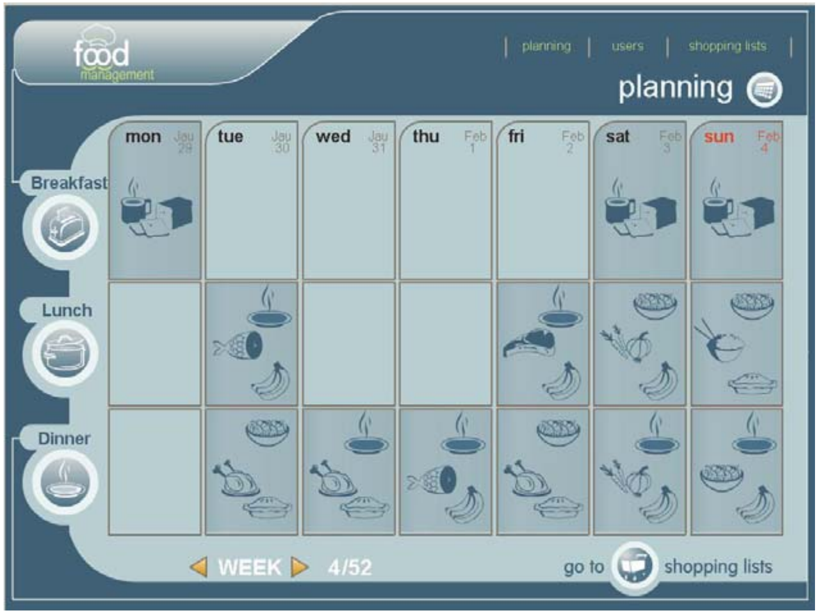
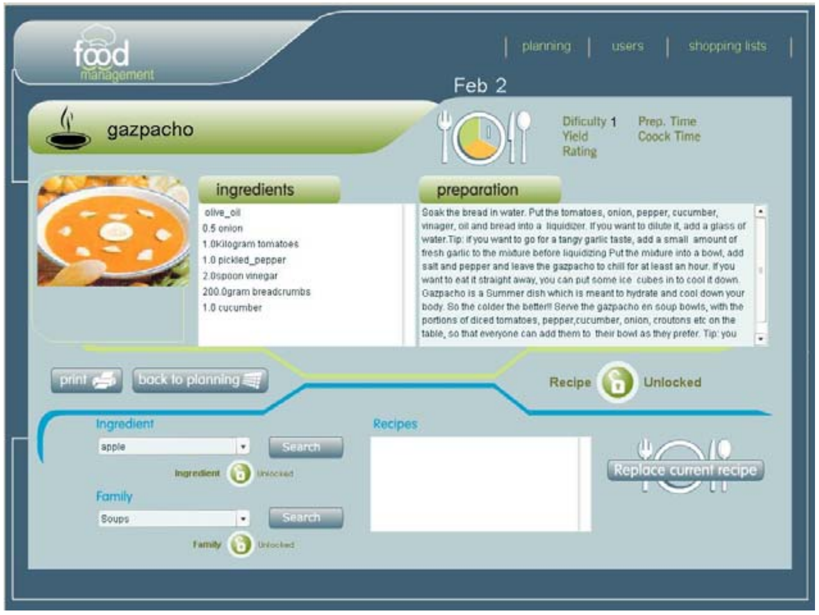


Figure 24 FM week suggestion

Detailed information can be seen if you click on any suggestion. You can see the starter, the main course or the dessert suggestion by clicking in the different coloured portions of the plate.

The user can change any suggestion manually. To do so, the user can search the recipes by ingredients or meat type (soup, meat, …).

The suggested recipes are saved in a XML file ordered by week number.

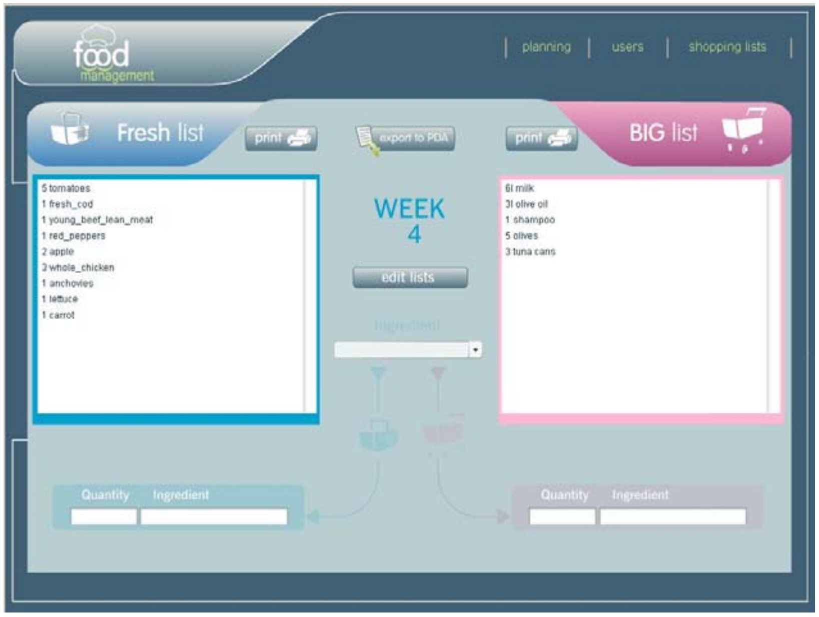


olive_oil
0.5 onion
1.0Kilogram tomatoes
1.0 pickled_pepper
2.0spoon vinegar
200.0gram breadcrumbs
1.0 cucumber

Soak the bread in water. Put the tomatoes, onion, pepper, cucumber, vinager, oil and bread into a liquidizer. If you want to dilute it, add a glass of water.Tip: if you want to go for a tangy garlic taste, add a small amount of fresh garlic to the mixture before liquidizing Put the mixture into a bowl, add salt and pepper and leave the gazpacho to chill for at least an hour. If you want to eat it straight away, you can put some ice cubes in to cool it down. Gazpacho is a Summer dish which is meant to hydrate and cool down your body. So the colder the better!! Serve the gazpacho en soup bowls, with the portions of diced tomatoes, pepper,cucumber, onion, croutons etc on the table, so that everyone can add them to their bowl as they prefer. Tip: you

Figure 25 FM day detail

The application will generate a Shopping List taking into account the ingredients that are actually in the fridge and the ingredients required to prepare the suggested recipes. A fresh shopping list is generated with fresh ingredients (to be consumed in the next few days, such as meat, lettuce,…) and another big list with non-perishable items, both of them for a week.



WEEK 4

5 tomatoes
1 fresh_cod
1 young_beef_lean_meat
1 red_peppers
2 apple
3 whole_chicken
1 anchovies
1 lettuce
1 carrot

6l milk
3l olive oil
1 shampoo
5 olives
3 tuna cans

Figure 26 FM shopping list screen

The user can add or delete ingredients from both lists. The ingredient is selected from the ingredient combo, then the user chooses if that element must be on the fresh list or on the big list.

After that, the user enters the quantity of that product that has to be added or deleted, clicking on the “add to list” button or the “delete” button.

The print and export to PDA functionalities are under development .

2.4 Health Management

2.4.1 Provider

Ikerlan

2.4.2 Development status

Functional prototype

2.4.3 Intended audience

Project partners who want or need to deploy and run the Health Management Application (HMA).

2.4.4 License

Proprietary.

2.4.5 Language

.Net C#

2.4.6 Environment info needed if you want to run this sw (service)

2.4.6.1 Hardware

Minimum requirements:

- A RFID user location system based on the HBL100 Wireless Network Controller from Sensite Solutions. This element is the advanced base station in the LogiSphere system. Wireless Network Controllers can be placed anywhere where the presence and status of Intelligent Tags needs to be detected and recorded. The detection range of the HBL100 can be configured from less than 1 metre to up to 300 metres.
- Intelligent tags BN 208 from Sensite Solutions.
- Wireless Weight Scale Medic4all with RF communication.
- Wireless Blood Pressure Monitor BP-102 with RF communication.
- ECG from Sension Digital Plaster with RF communication.
- Microx Pulsoximeter with Bluetooth communication.
- A PC with communication resources for Power Line, WiFi and RF.

2.4.6.2 Software

- Windows PC Software
- .Net Framework 2.0
- AMIGO Middleware
- OWLDotNetApi (OWL parser)
- NxBRE libraries (rule engine)

- Flash.ocx (integrate Flash Interfaces on Applications)
- Java Runtime Environment 1.5
- Medical4all software for health measure adquisition
- MD – Gateway – Helper to create all Context Sources
- User location application based on RFID.
- Health Management Application

2.4.7 Platform

Any system capable of running the software requirements (.Net, Windows).

2.4.8 Files

Available in GForce

2.4.9 Documents

This document provides information about the Health Management Application. Also a tutorial is provided in the next sections.

General description of Health Management Application, its architecture and relation to other Amigo components are given in documents D5.1, D5.2 and D5.3.

2.4.10 Tasks

Integration with other components from AMIGO system to develop a fully functional application as described in document D5.2.

2.4.11 Bugs

None so far.

2.4.12 Patches

None so far.

2.4.13 Deployment

2.4.13.1 System requirements

System requirements exclusively depend on the features of the application. Thus, a typical platform with the following minimum requirements will be needed:

- Pentium 1 GHz or equivalent processor
- 512 MB of RAM
- 25 MB of disk space
- Minimal Screen resolution must be 102x768 (full screen)

For System requirements of other AMIGO components needed to run Health Management Application, please refer to the appropriate documentation.

2.4.14 Download

Files required to run the Health Management Application can be downloaded from the Inria GForce repository available at:

/svn+ssh://login@scm.gforce.inria.fr/svn/amigo

2.4.15 Installation

Health Management Application is provided as a software application ready to deploy on a .NET platform. It is being developed an specific application to install the Health Management Application in the AMIGO environment.

2.4.16 Configuration

2.4.16.1 Application configuration

The following AMIGO bundles in OSCAR are needed:

0	Active	System Bundle
1	Active	Shell Service
2	Active	Table Layout
3	Active	Shell GUI
4	Active	Shell Plugin
5	Active	Bundle Repository
6	Active	log4j
7	Active	Service Binder
8	Active	amigo_core
9	Active	amigo_ksoap_binding
10	Active	Servlet
11	Active	HTTP Service (+ Amigo mods)
12	Active	amigo_ksoap_export
13	Active	amigo_wsdiscovery
14	Active	jena-2.4
15	Active	context-broker-service
16	Active	Context Source Manager
17	Active	Context Helper

The application loads the configuration variables from the user profiles (UMPS) and the historics .xme from the historic file next to the healthmanagement.exe.

To have RF communications working correctly, the RF driver provided by Medic4all have to be installed.

In the cmconfig.xml file the port that support the RFID communication has to be specified.

The Sensiun driver needs to be installed in order to get the ECG values.

2.4.16.2 Network configuration

In order to let the application work properly, it is convenient that every modules are installed in the same platform.

2.4.16.3 Compiling

Not applicable. All packages are precompiled.

2.4.17 Integrated applications

From WP3, AMIGO Middleware based on .Net has been used.

From WP4: Context Sources from CMS (health devices and location); UMPS (user's preferences); UIS (visual); ANS (behaviour rule definition).

2.4.18 User Guides

The Health Management application is based on the home having a specific space dedicated to monitoring physiological values. On that basis, it helps look after the health of those using it, by carrying out certain tasks involving information processing, trend checking and informing the users about any significant changes that need to be looked at by healthcare professionals.

In the Home Care and Safety demonstrator, this space has been installed in the bathroom, where there are systems to monitor weight, body temperature, blood pressure, heart rate, oxygenation of the blood and to do an ECG.

Each user has an active RFID identifier which will enable the AMIGO middleware to know who is at home and the specific location of each person, so that the Health Management application knows who is using the physiological monitoring resources, thereby enabling the pick-up, storage, analysis and notification processes, if required, to be personalised.

The aim is for the RFID identifiers to be incorporated into the clothes people wear at home, so that having to carry them with you does not become awkward or uncomfortable.

The following figures show the systems used to pick up the physiological values.

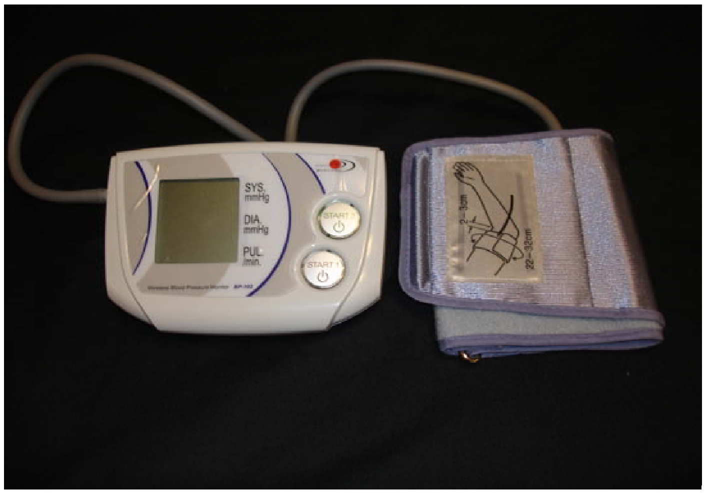


Figure 27 Wireless Blood Pressure Monitor BP-102 with RF communication

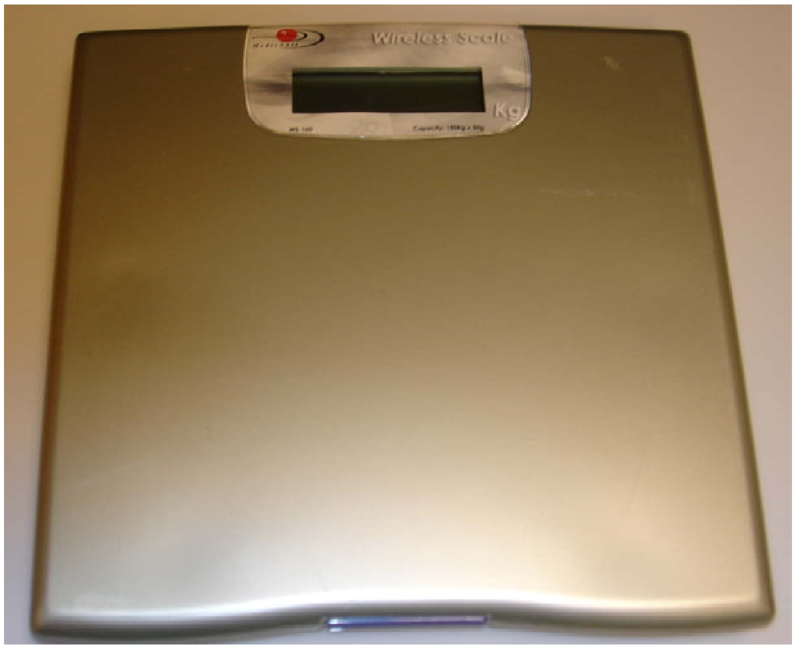


Figure 28 Wireless Weight Scale Medic4all with RF communication

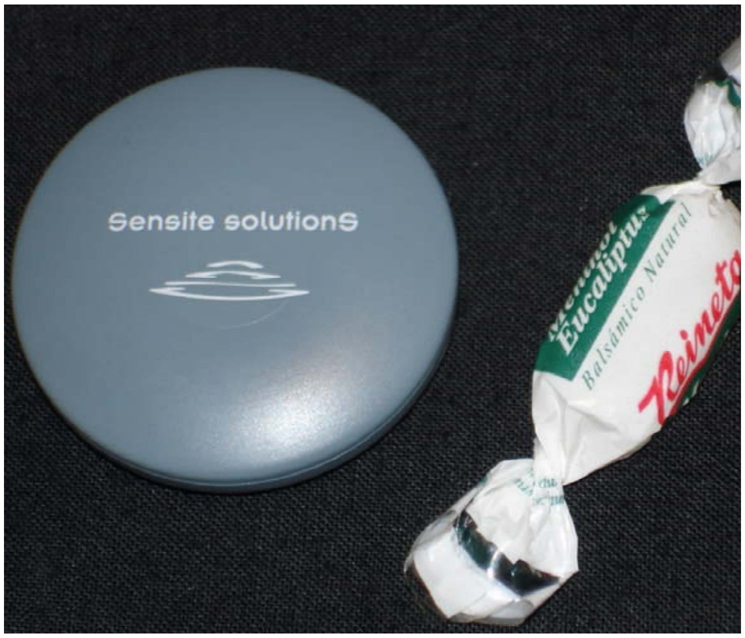


Figure 29 Intelligent tags BN 208 from Sensite Solutions

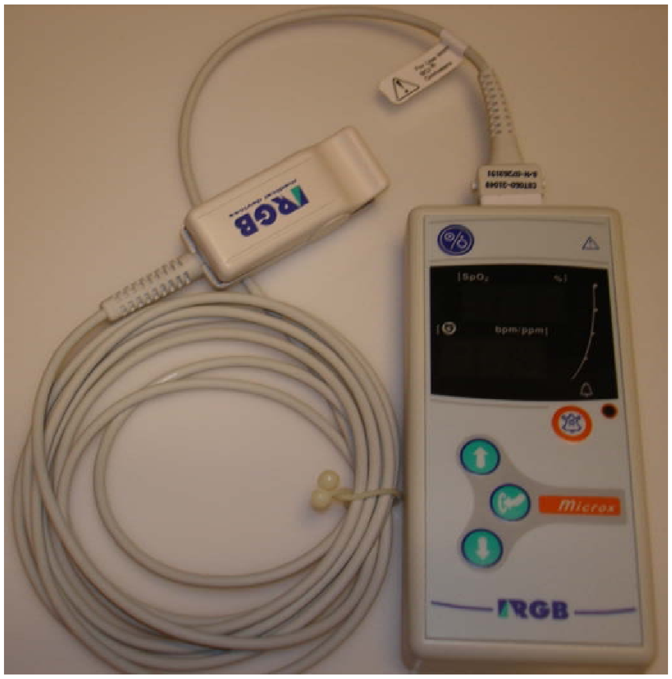


Figure 30 Microx Pulsoximeter with Bluetooth communication

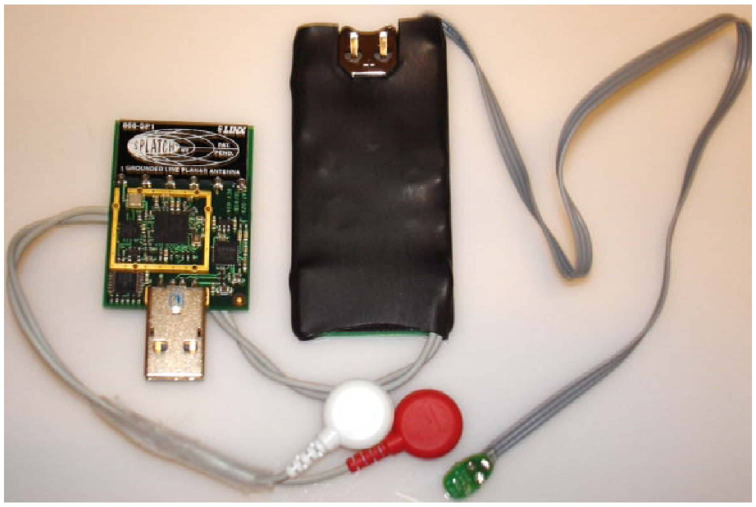


Figure 31 ECG from Sensiun Digital Plaster with RF communication

Users can use these systems whenever, to the extent and in the order they prefer, as the systems are not subject to any procedure or protocol for their use, except for the obligation of users having to wear an identification tag.

The use of any of these systems will activate the Health Management application, which will store the data received in the historic records for the user in question and show the performance of the value measured using the AMIGO middleware interface resources.

If any problem is detected in the data analysis, especially any significant variation with regard to the trend for that value, the user will be informed.

The following figures show the modes for the Health Management application's relationship with the user.

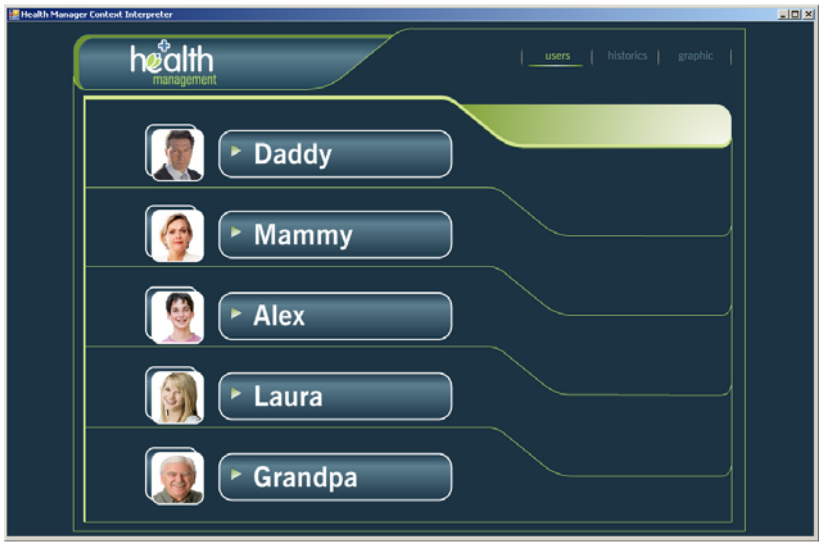


Figure 32 HM user list

Each user can visualize all health measurements and graphical evolution

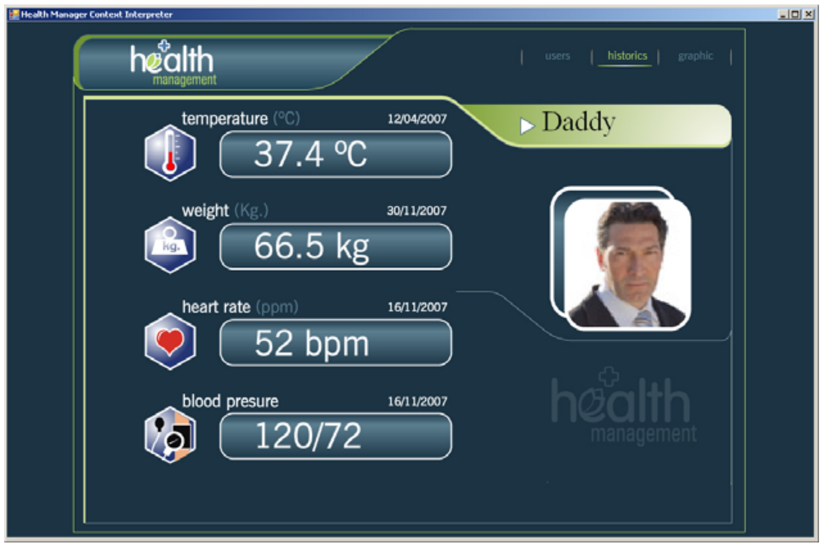


Figure 33 HM collected data

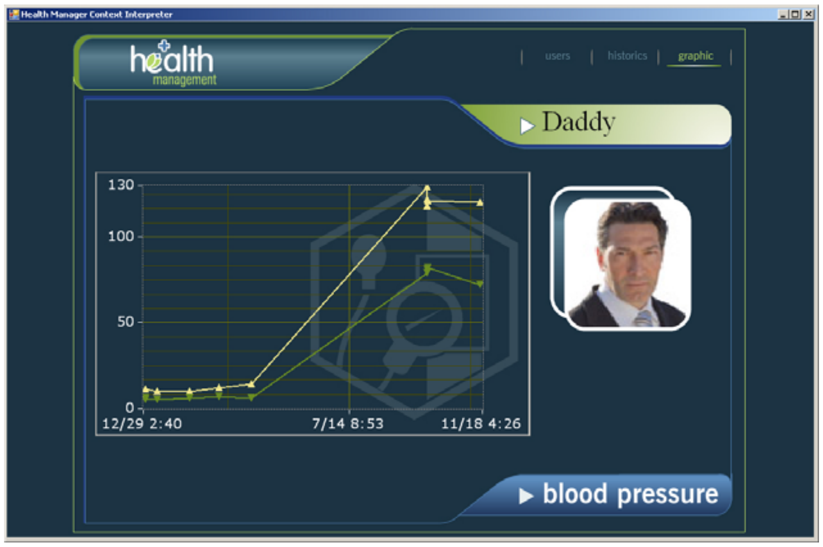


Figure 34 HM graphic screen

When the user starts collecting data from the ECG will appear the following screen:

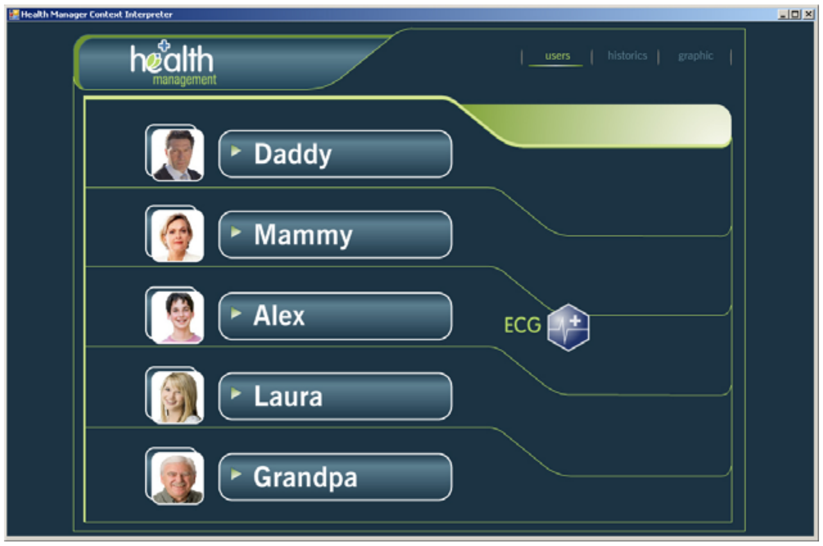

Health Manager Context Interpreter
health management
users | historics | graphic
Daddy
Mammy
Alex
Laura
Grandpa
ECG

Figure 35 HM collecting ECG data

The user can also send data to a health data.

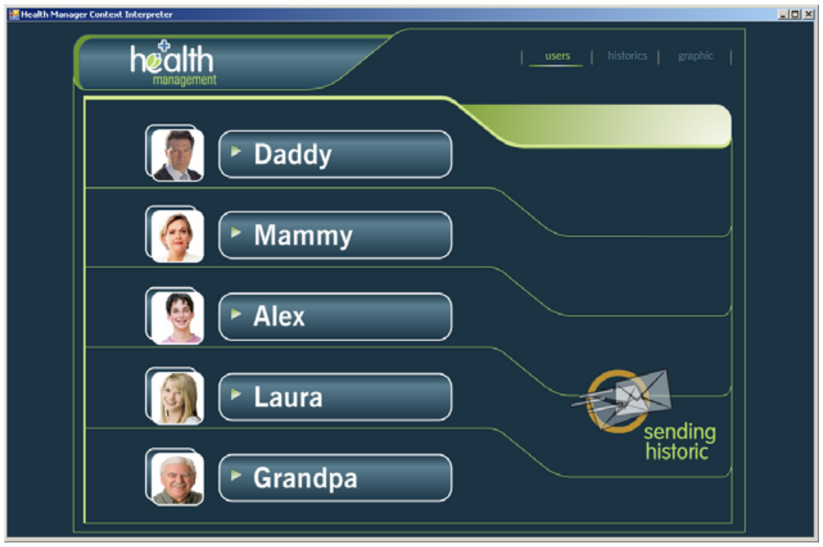


Figure 36 HM sending data to health center

2.5 Appliances Management

2.5.1 Provider
Ikerlan

2.5.2 Development status
Functional prototype

2.5.3 Intended audience
Project partners who want or need to deploy and run the Appliances Manager application.

2.5.4 License
Proprietary.

2.5.5 Language
.Net C#

2.5.6 Environment info needed if you want to run this sw (service)

2.5.6.1 Hardware
Minimum requirements:

- The Fagor Home Automation Network:
  - Domotic washing-machine
  - Domotic dishwasher
  - Domotic oven
  - Domotic boiler
  - Domotic fridge
  - Domotic induction hob
- A PC with communication resources for Power Line, WiFi and RF.

2.5.6.2 Software
- Windows PC Software
- .Net Framework 2.0
- AMIGO Middleware
- OWLDotNetApi (OWL parser)
- NxBRE libraries (rule engine)
- Flash.ocx (integrate Flash Interfaces on Applications)
- Java Runtime Environment 1.5
- MD – Gateway – Helper to create all Context Sources

- Appliances Manager Application

2.5.7 Platform

Any system capable of running the software requirements (.Net, Windows).

2.5.8 Files

Available in GForce

2.5.9 Documents

Next sections of this document provide information about the Appliances Manager Application. Also a tutorial is provided in the next sections.

General description of Appliances Manager Application, its architecture and relation to other Amigo components are given in documents D5.1, D5.2 and D5.3.

2.5.10 Tasks

Integration with other components from AMIGO system to develop a fully functional application as described in document D5.2.

2.5.11 Bugs

None so far.

2.5.12 Patches

None so far.

2.5.13 Deployment

2.5.13.1 System requirements

System requirements exclusively depend on the features of the application. Thus, a typical platform with the following minimum requirements will be needed:

- Pentium 1 GHz or equivalent processor
- 512 MB of RAM
- 25 MB of disk space
- Minimal Screen resolution must be 102x768 (full screen)

For system requirements of other AMIGO components needed to run Appliances Manager Application, please refer to the appropriate documentation.

2.5.14 Download

Files required to run the Appliances Manager Application can be downloaded from the Inria GForce repository available at:

/svn+ssh://login@scm.gforce.inria.fr/svn/amigo

2.5.15 Installation

Appliances Manager Application is provided as a software application ready to deploy on a .NET platform. It is being developed an specific application to install the Appliances Manager Application in the AMIGO environment that will be use of the Gateway application under \ius\context_mgmt\household_appliance.

2.5.16 Configuration

This application loads many configuration files, but the appliances related one is located in the Config directory(DCConfig.xml).

2.5.16.1 Application configuration

The following AMIGO bundles in OSCAR are needed:

0	Active	System Bundle
1	Active	Shell Service
2	Active	Table Layout
3	Active	Shell GUI
4	Active	Shell Plugin
5	Active	Bundle Repository
6	Active	log4j
7	Active	Service Binder
8	Active	amigo_core
9	Active	amigo_ksoap_binding
10	Active	Servlet
11	Active	HTTP Service (+ Amigo mods)
12	Active	amigo_ksoap_export
13	Active	amigo_wsdiscovery
14	Active	jena-2.4
15	Active	context-broker-service
16	Active	Context Source Manager
17	Active	Context Helper
18	Active	Context Source UPnP Devices
19	Active	Context Source Client UPnP Devices
20	Active	Context Source Health Data

The application loads the configuration variables from the CookingAssistant.exe.config file, such as user profile, output directory for the week planning, etc.

2.5.16.2 Network configuration

In order to let the application work properly, it is convenient that every modules are installed in the same platform.

2.5.16.3 Compiling

Not applicable. All packages are precompiled.

2.5.17 Integrated applications

From WP3, AMIGO Middleware based on .Net has been used.

From WP4: Context Sources from CMS (fridge); UMPS (user's preferences); UIS (visual).

2.5.18 User Guides

The Appliances Manager application manages the set of domestic appliances assigned to the Fagor Home Automation Network. These appliances have been fitted with communications resources via the Power Line (phsyical driver and Fagor Bus communiations protocol).

By means of this application, users have, regardless of where they are, absolute control over their domestic appliances to switch them on, turn them off or know what their status is. If there is any problem with how they are working, the appliances will inform Fagor's technical service.

2.5.18.1 Actions of the Appliances Manager application on the washing machine

Figure 37 Domotic washing-machine

- Remote start / stop of the wash programme
- Remote programming of the wash programme
- Operation status enquiry
- Operation by priorities

- Telediagnosis:
  - Overflowing
  - Water heating fault
  - Temperature sensor fault
  - Fault in the programme selector
  - Breakdown in the motor
  - Door open
  - Does not fill with water or empty it out
  - Unbalanced load in the drum

2.5.18.2 Actions of the Appliances Manager application on the dishwasher

Figure 38 Domotic dishwasher

- Remote start / stop of the wash programme
- Remote programming of the wash programme
- Operation status enquiry
- Operation by priorities
- Telediagnosis:
  - Overflowing
  - Water heating fault
  - Temperature sensor fault
  - Water pressure fault
  - Door open
  - Door safety locking fault

- Does not fill with water or empty it out
- No salt
- No rinse aid

2.5.18.3 Actions of the Appliances Manager application on the fridge

Figure 39 Domotic fridge

- Remote action of fast refrigeration function
- Remore activation of fast freeze function
- Remote switching off of the holiday function
- Operation status enquiry
- Telediagnosis:

  - Break in the cold chain in the freezer
  - Breakdown in the sensors
  - Refrigerator door open
  - Freezer door open

2.5.18.4 Actions of the Appliances Manager application on the oven

Figure 40 Domotic oven

- Remote start / stop of the cooking programme
- Remote programming of the cooking programme
- Operation status enquiry
- Operation by priorities
- Telediagnosis:
    - Fault in the temperature sensor

2.5.18.5 Actions of the Appliances Manager application on the boiler (central heating):

Figure 41 Domotic boiler

- Remote start / stop
- Remote programming
- Operation status enquiry
- Telediagnosis:
    - No water and/or gas
    - Fault in the safety thermostat
    - Burner outlet water temperature too high
    - Fault in the gas valve relay
    - Fault in the temperature sensors
    - Fault in the fume removal system
    - Fault in the water circuit
    - Fault in the control card
    - Fault in the flame detection

2.5.18.6 Actions of the Appliances Manager application on the induction hob

Figure 42 Domotic induction hob

- Remote switching off of the hotplates
- Operation status enquiry
- Operation by priorities

2.6 Comfort System

2.6.1 Provider

Ikerlan

2.6.2 Development status

Functional prototype

2.6.3 Intended audience

Project partners who want or need to deploy and run the Comfort System application.

2.6.4 License

Proprietary.

2.6.5 Language

.Net C#

2.6.6 Environment info needed if you want to run this sw (service)

2.6.6.1 Hardware

Minimum requirements:

- From the Fagor Home Automation Network:
  - Several temperature sensors with Bluetooth resources.
  - Several actuators connected to Power Line. These devices will activate and deactivate the electric heaters.
  - A special object designed to show the comfort level by means of the colour of a set of led diodos included in it. This object is part of the Fagor Home Automation Network and has got communication resources for Power Line. The idea is to use decorative devices for Ambient Intelligence interface functions. This element is called Sense_Object.
- A PC with communication resources for Power Line, WiFi and RF.

2.6.6.2 Software

- Windows PC Software
- .Net Framework 2.0
- AMIGO Middleware
- OWLDotNetApi (OWL parser)
- NxBRE libraries (rule engine)
- Flash.ocx (integrate Flash Interfaces on Applications)
- Java Runtime Environment 1.5

- MD – Gateway – Helper to create all Context Sources
- Comfort System Application

2.6.7 Platform

Any system capable of running the software requirements (.Net, Windows).

2.6.8 Files

Available in GForce

2.6.9 Documents

Next sections of this document provide information about the Comfort System Application. Also a tutorial is provided in the next sections.

General description of Comfort System Application, its architecture and relation to other Amigo components are given in documents D5.1, D5.2 and D5.3.

2.6.10 Tasks

Integration with other components from AMIGO system to develop a fully functional application as described in document D5.2.

2.6.11 Bugs

None so far.

2.6.12 Patches

None so far.

2.6.13 Deployment

2.6.13.1 System requirements

System requirements exclusively depend on the features of the application. Thus, a typical platform with the following minimum requirements will be needed:

- Pentium 1 GHz or equivalent processor
- 512 MB of RAM
- 25 MB of disk space
- Minimal Screen resolution must be 102x768 (full screen)

For system requirements of other AMIGO components needed to run Comfort System Application, please refer to the appropriate documentation.

2.6.14 Download

Files required to run the Appliances Manager Application can be downloaded from the Inria GForce repository available at:

/svn+ssh://login@scm.gforce.inria.fr/svn/amigo

2.6.15 Installation

Comfort System Application is provided as a software application ready to deploy on a .NET platform. It is being developed an specific application to install the Comfort System Application in the AMIGO environment that will be use of the Gateway application under \ius\context_mgmt\household_appliance.

2.6.16 Configuration

This application loads many configuration files, but the appliances related one is located in the Config directory(TSConfig.xml).

2.6.16.1 Application configuration

The following AMIGO bundles in OSCAR are needed:

0	Active	System Bundle
1	Active	Shell Service
2	Active	Table Layout
3	Active	Shell GUI
4	Active	Shell Plugin
5	Active	Bundle Repository
6	Active	log4j
7	Active	Service Binder
8	Active	amigo_core
9	Active	amigo_ksoap_binding
10	Active	Servlet
11	Active	HTTP Service (+ Amigo mods)
12	Active	amigo_ksoap_export
13	Active	amigo_wsdiscovery
14	Active	jena-2.4
15	Active	context-broker-service
16	Active	Context Source Manager
17	Active	Context Helper
18	Active	Context Source UPnP Devices
19	Active	Context Source Client UPnP Devices
20	Active	Context Source Health Data

The application loads the configuration variables from the CookingAssistant.exe.config file, such as user profile, output directory for the week planning, etc.

2.6.16.2 Network configuration

In order to let the application work properly, it is convenient that every modules are installed in the same platform.

2.6.16.3 Compiling

Not applicable. All packages are precompiled.

2.6.17 Integrated applications

From WP3, AMIGO Middleware based on .Net has been used.

From WP4: Context Sources from CMS (fridge); UMPS (user's preferences); UIS (visual).

2.6.18 User Guides

The Comfort System application will have two temperature sensors which will connect with the AMIGO middleware via Bluetooth. These sensors are responsible for measuring the ambient temperature in two different zones of the home.

In each of these zones, there will be an automatism, connected to the AMIGO middleware via the Power Line, which is in charge of switching an electric heater on and off.

Thanks to these resources, the Comfort System application will have a “zoning” feature for generating heat in the home (zoning: having a number of different zones in the home as far as ambient temperature regulation is concerned).

One of the zones, presumably the one that includes the lounge, will contain the user interface and decorative element known as the Sense_Object, which will reflect the temperature level in the zone using a range of colours from deep blue (cold) to red (warm). The Sense_Object will communicate with the Comfort System application via the Power Line.

The user can set a temperature or profile for each zone, using the different interface resources available in the AMIGO middleware. A profile is a list of temperatures and timebands, so that a certain temperature can be programmed for a specific time of day, normally taking into account the tmes when there is no-one at home. These profiles can be different depending on whether the day is a holiday or a normal working day.

The comfort sensors will periodically send the application the temperature for the zone they are located in, using their Bluetooth link. On the basis of this data, the application will determine whether the electric radiators have to be switched on or off, sending the relevant signal down the Power Line.

Whilst the Comfort System application is active, it will send the zone temperature data to the Sense_Object, so that this element can reflect it by means of the relevant colour.

The user can change the setting or profiles for each of the zones at any time.

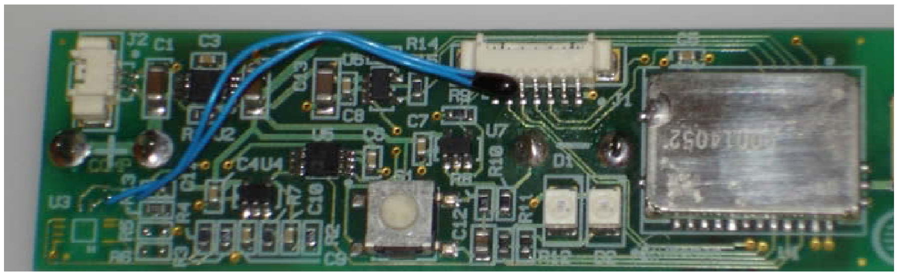


Figure 43 Sensor with Bluetooth communication

2.7 Technical Alarms Management

2.7.1 Provider
Ikerlan

2.7.2 Development status
Functional prototype

2.7.3 Intended audience
Project partners who want or need to deploy and run the Technical Alarm application.

2.7.4 License
Proprietary.

2.7.5 Language
.Net C#

2.7.6 Environment info needed if you want to run this sw (service)

2.7.6.1 Hardware
Minimum requirements:

- The Fagor Home Automation Network:
    - Domotic washing-machine
    - Domotic dishwasher
    - Domotic water sensor
    - Domotic gas sensor
    - Domotic water actuator
    - Domotic gas actuator
- A PC with communication resources for Power Line, WiFi and RF.

2.7.6.2 Software
- Windows PC Software
- .Net Framework 2.0
- AMIGO Middleware
- OWLDotNetApi (OWL parser)
- NxBRE libraries (rule engine)
- Flash.ocx (integrate Flash Interfaces on Applications)
- Java Runtime Environment 1.5
- Medical4all software for health measure adquisition

- MD – Gateway – Helper to create all Context Sources
- Technical Alarm Application

2.7.7 Platform

Any system capable of running the software requirements (.Net, Windows).

2.7.8 Files

Available in GForce

2.7.9 Documents

Next sections of this document provide information about the Technical Alarm Application. Also a tutorial is provided in the next sections.

General description of Technical Alarm Application, its architecture and relation to other Amigo components are given in documents D5.1, D5.2 and D5.3.

2.7.10 Tasks

Integration with other components from AMIGO system to develop a fully functional application as described in document D5.2.

2.7.11 Bugs

None so far.

2.7.12 Patches

None so far.

2.7.13 Deployment

2.7.13.1 System requirements

System requirements exclusively depend on the features of the application. Thus, a typical platform with the following minimum requirements will be needed:

- Pentium 1 GHz or equivalent processor
- 512 MB of RAM
- 25 MB of disk space
- Minimal Screen resolution must be 102x768 (full screen)

For system requirements of other AMIGO components needed to run Technical Alarm Application, please refer to the appropriate documentation.

2.7.14 Download

Files required to run the Technical Alarm Application can be downloaded from the Inria GForce repository available at:

/svn+ssh://login@scm.gforce.inria.fr/svn/amigo

2.7.15 Installation

Technical Alarm Application is provided as a software application ready to deploy on a .NET platform. It is being developed an specific application to install the Technical Alarm Application in the AMIGO environment.

2.7.16 Configuration

2.7.16.1 Application configuration

The following AMIGO bundles in OSCAR are needed:

0	Active	System Bundle
1	Active	Shell Service
2	Active	Table Layout
3	Active	Shell GUI
4	Active	Shell Plugin
5	Active	Bundle Repository
6	Active	log4j
7	Active	Service Binder
8	Active	amigo_core
9	Active	amigo_ksoap_binding
10	Active	Servlet
11	Active	HTTP Service (+ Amigo mods)
12	Active	amigo_ksoap_export
13	Active	amigo_wsdiscovery
14	Active	jena-2.4
15	Active	context-broker-service
16	Active	Context Source Manager
17	Active	Context Helper
18	Active	Context Source UPnP Devices
19	Active	Context Source Client UPnP Devices
20	Active	Context Source Health Data

The application loads the configuration variables from the CookingAssistant.exe.config file, such as user profile, output directory for the week planning, etc.

2.7.16.2 Network configuration

In order to let the application work properly, it is convenient that every modules are installed in the same platform.

2.7.16.3 Compiling

Not applicable. All packages are precompiled.

2.7.17 Integrated applications

From WP3, AMIGO Middleware based on .Net has been used.

From WP4: Context Sources from CMS (fridge); UMPS (user's preferences); UIS (visual).

2.7.18 User Guides

Gas and water leaks are one of the most delicate problems to deal with in the home. The aim of Fagor's technical alarms system, integrated in the AMIGO middleware project, is to safeguard homes from these problems, by means of set of detectors and actuators which keep a look out for any possible problems.

The detectors or sensors detect any leaks that may occur, either water or gas leaks, whilst the actuators close the relevant throttle valve.

The process is as follows, taking a water leak as an example: there is a network of sensors and actuators linked to the Technical Alarm application via the Power Line. If any of the sensors detects water for a few seconds (the aim here is to prevent false alarms), confirming that there is a leak, the sensor involved sends the relevant message to the application, down the Power Line. Immediately, the Technical Alarm application sends, also down the Power Line, an order to close the electrically-operated valves to all the water actuators that the safety network has.

Simultaneously, the Technical Alarm application will locally inform the user of this contingency, using for this purpose the different user interface resources available on the AMIGO platform.

Finally, if by means of the user location system the application knows that no-one is at home, the remote notification service will be activated.

Once the cause of the leak has been cleared up, the user has to tell the application by pressing the presence button of the sensor that detected the leak.

Once the Technical Alarm application has received from the sensor the information that the physical value detected earlier has disappeared and that the user is now at home, the application will send down the Power Line the order to open the electrically-operated valves to the actuators that it had previously closed.

The elements involved in the process are:

Figure 44 Water leak or flood detector

Flood detector ready to detect leaks by means of a sensor at floor level, which is equipped with output signals to activate a shut-off valve and send the alarm to the Fagor Maior Domo®.

Functions

Flood detection by means of the floor-level sensor

Acoustic and luminous alarm signal

Communication to the Maior-Domo® of the alarms detected

Characteristics

PowerLine communication with birectional Fagor Bus

Frequency: 132.45 KHz

Supply: 230 Vac

Shut-off valve output: 230 Vac (max. 60W).

Acoustic alarm signal: 85 dBA at 3m.

Installation in 60 x 60 mm universal box.

Figure 45 Electrically-operated water valve control

Home automation control for electrically-operated water valve whose function is to order the closure of the valve located at the main water inlet when an alarm is received from the detectors.

Functions

Controls an electrically-operated valve of the Normally Open type to cut off the water supply.

Electrically-operated-valve-activated luminous signal.

Characteristics

PowerLine communication with birdirectional Fagor Bus

Frequency: 132.45 KHz

Supply: 230 Vac

Signal output for electrically-operated valve wired connection: 230 Vac (max. 60W).

Installation in 60 x 60 mm universal box.

Figure 46 Gas detector

Gas detector ready to detect Liquid Petroleum gases (DG-400 GLP B) or Natural Gas (DG-400 N B) which is equipped with output signals to activate a shut-off valve and communicate the alarm to the Fagor Maior-Domo®.

Functions

Detection of Liquid Petroleum Gases (butane, propane) or Natural Gas.

Acoustic and luminous alarm signal

Communication to the Maior-Domo® of the alarms detected

Characteristics

PowerLine communication with bidirectional Fagor Bus

Frequency: 132.45 KHz

Supply: 230 Vac

Shut-off valve output: 230 Vac (max. 60W).

Acoustic alarm signal: 85 dBA at 3m.

Installation in 60 x 60 mm universal box.

Figure 47 Control for has electrically-operated gas valve

Home automation for electrivally-operated gas valve whose function is to order the closure of the valve on the main gas inlet when an alarm is received from the detectors.

Functions

Controls an electrically-operated valve of the Normally Open type to cut off the gas supply

Electrically-operated-valve-activated luminous signal

Characteristics

PowerLine communication with bidirectional Fagor Bus

Frequency: 132.45 KHz

Supply: 230 Vac

Signal output for electrically-operated valve wired connection: 230 Vac (max. 60W).

Installation in 60 x 60 mm universal box.

2.8 Entrance Manager

The Entrance Manager manages the front door events. Entrance Manager is responsible for recognizing people at the front and patio doors of the house and opening the door(s) for them depended on their authorization for such action. Special situations can affect the behaviour of the entrance manager, for example if Amigo has detected an emergency (like an elderly person has fallen and can't get up again), then the door should also be opened for e.g. ambulance personnel.

The Entrance Manager application is divided in two sub-applications: Allow the entrance and Get visitor's message, this last one is strongly related with Messaging board application WP6 [Amigo-WP6IniSpec].

2.8.1 Provider

SingularLogic

2.8.2 Development status

First version for .NET implemented.

2.8.3 Intended audience

Project partners

2.8.4 License

The software itself will be under LGPL license, but might make use of proprietary binaries/ libraries for which no source code is provided.

2.8.5 Language

C#

2.8.6 Environment (set-up) info needed if you want to run this sw (service)

2.8.6.1 Hardware

- Microphone and loudspeakers for speech input/output.
- Automatic door operator (lock and open/close)
- Camera to be used by the Face Recognition Context Source
- Bluetooth phone and sensor (+Context Source) as an input for the Face Recognition Context Source to improve the reliability (optional)

2.8.6.2 Software

- Windows 2K
- Amigo .NET 2.0 Based Programming & Deployment Framework from WP3
- Amigo services: UMPS, UIS (specifically, VoiceIO and SpeakerRecognition services), CMS (Context Broker and Face Recognition Context Source)
- Microsoft SQL Server
- Other Amigo applications: Messaging Board (WP6)

2.8.7 Platform

Microsoft .NET Framework v2.0

Microsoft Visual Studio 2005

2.8.8 Files

2.8.9 Documents

Application architecture is described in D5.2.

2.8.10 Tasks

Implementation of needed context sources.

Complete integration with UMPS, CMS, IUS.

2.8.11 Bugs

-

2.8.12 Patches

-

2.8.13 Deployment

2.8.13.1 System requirements

Amigo .NET programming framework

Microsoft .NET 2.0 Framework v2.0

SQL Server

UMPS service

VoiceIO service

Speaker Recognition Service

MessagingBoard Application

CMS - Context Broker

Face Recognition Context Source

2.8.14 Download

Not applicable.

2.8.15 Installation

The application itself is an executable, and doesn't need to be installed. However, before running the application all other dependencies (services and applications used) should be installed and configured. The application specific database should be extracted (from backup) on the SQL server to be used at runtime.

2.8.16 Configuration

In the "EntranceManager.ini" file you can set:

- ApplicationPath – the path where the application is installed;

- DefaultLanguage - to be used if no UMPS is available or if the PrefferedLanguage of the user is not between the available languages;

- AvailableLanguages – all languages for which there are valid language resources implemented (output messages and grammar resources);

- DBConnection – specify the SQL Server where the application specific database is located, the user name and password to be used for connection, as well as the name of the database (if changed).

If the application is extended to other languages than en-US, for which language resources are provided, the corresponding language resources should be set.

2.8.17 Integrated applications

The Entrance Manager sub-application uses of the following services: UMPS, VoiceIO service, Speaker Recognition service and Context Management service. If the VoiceIO service is not available the application cannot run, since the user-system interaction is based on speech interface.

2.8.17.1 VoiceIO service integration

This service is used by the Entrance Manager sub-application to perform all interactions (input-output) between the user and the application. Specific application resources, grammars and prompts, have been implemented for each interaction node. These resources are language dependents, and the implemented ones are for American English (en-US).

2.8.17.2 UMPS integration

The Entrance Manager sub-application is using this service for the following situations:

- to generate/update language resources

- to personalize audio output

- to check claimed relationship of person at front door with family members

- to verify entrance rights for the person at front door (if other than family members).

For the generation of language resources and personalization of audio output, the sections to be filled using UMPS are explicitly described in the corresponding files. Following is an example of a rule from a grammar template:

```
<rule id = "onlyLastName">
        <tag>LastName="; userID="</tag>
        <one-of>
<item><umps>PersonalDetails:LastName</umps><tag>LastName=<umps>PersonalDetails:LastName</umps>; userID=<umps>userID</umps></tag></item>
        </one-of>
</rule>
```

In this example, the statment <umps>PersonalDetails:LastName</umps> indicates which user profile setting will be used to fill the rule. For each user recorded in the DB of UMPS, a new item will be added to the rule. Grammars must be generated before running the application.

In a similar way, in order to produce personalized output, the prompt template has the form:

"<umps>PersonalDetails:FirstName</umps>, please wait, I will open the door for you!". In this case, UMPS is called at runtime, and the final audio output will be "Jerry, please wait, I will open the door for you!".

In order to allow the entrance of a person in the house, the UMPS setting "SecurityRights:HouseEnter" should be set to 'enabled'.

2.8.17.3 Speaker Recognition service integration

This service can be used only if a voice print (speaker model) has been previously generated and trained. This should be done for each user that is allowed to enter the house after identification and verification with the Entrance Manager sub-application. Training should be performed in the conditions (noise), and with similar hardware devices (microphone), that will be valid also at runtime, when recognition must be performed.

To avoid security issued that could occur if intrusion is aimed with recorded voice of an authorized person, the person at front door is asked to say a 4-digits code, which is random-generated at runtime. Better results can be obtained with longer phrases, but this would increase the waiting time. Also, if there are more than 4 digits, the risk that the person at front door forgets the digits to be spoken appears.

2.8.17.4 Context Management service/ Face Recognition Context Source integration

The Context Broker component of CMS is used to:

- access the Face Recognition context source.

- retrieve information of persons presence/location in the house from any available context sources.

Person detection and identification at front door is performed by the Face Recognition context source. The difference between simple detection and the actual identification of the person is given by the confidence in the recognition result returned by the context source. If the confidence is low, then the person is considered as unknown for the system. The confidence limit is established for each hardware and environment conditions (camera, illumination). Similar to the Speaker Recognition service, a previously generated and trained model must be available for each user that the system should identify at front door.

2.8.18 User guides

2.8.18.1 Introduction

This section describes the functionality of the Entrance Manager application and possible problems that can appear during its usage.

2.8.18.2 Entrance Manager as a service

This application, except of its basic functionality to interact with persons at front door, is also running as a service itself, offering to other applications and services the possibility to allow entrance in the house by setting temporary, password-protected, entrance rights.

Once the EntranceManager.exe is started, it runs as a service. First of all, if the Face Recognition context source is available, it register to be informed by this context source whenever a person is detected by the video camera, and the interaction with the detected person starts in this case.

Otherwise, if the Face Recognition context source is not available, but there is another application or service that detects persons at front door, a web method is available to start the interaction:

IEntranceManagement::personArrived(string userID)

If the userID is empty, the person is considered unknown.

Except of persons for whom explicit entrance rights have been set in their profile, persons can also be allowed to enter by setting time-limited and password-protected entrance rights. Applications and services can set such rights by calling the following web method:

IEntranceManagement::setEntranceRight(string serviceID, string sessionID, string userID, string startTime, string endTime, string password, int useCount)

where:

- serviceID – the identification name of the service trying to set a temporary entrance right.

- sessionID - the identification name of the sessions which called the web method.

- userID – the identification name of the user for which the entrance right is valid. No empty string is allowed. If the right is valid for any user that has the valid password, then the value should be "ALL".

- startTime and endTime – define the temporary interval for which the password is valid. These parameters should provide both, date and time information (i.e. 21/07/2007 14:30:21). The temporary interval should be smaller than 24 hours.

- password – the verification string, which should contain only numeric characters. The person trying to use this access right will be asked to speak the digits.

- useCount – an integer indicating how many times the password can be used. If the value is -1, then the password is set for multiple uses. If the value equal to zero, the password is invalid. Values less than zero, other than -1 are not allowed.

Every time the application starts the interaction with a user at front door, a database procedure which retrieves all passwords valid at that time is called, and if applicable the user is asked to supply its password.

2.8.18.3 Interaction Resources

The user-system interaction is speech-based. For this interaction, language specific resources, grammars and prompts, have been implemented.

The most complex grammar is the "../ApplicationResources/Grammars/en-US/EM_main.grxml", as at the first interaction of the application with the user at front door there are many parameters which can be filled at the same time:

- the "Scope" of the presence at front door, with currently implemented possible values ENTER || DELIVERY || EMERGENCY || VISIT.

- the "userID" parameter, which is filled if the scope is ENTER or VISIT. This parameter is optional, as the visitor may identify himself or not. The possible values for this parameter are given by reference to an external grammar, ""../ApplicationResources/Grammars/en-US/Users.grxml".

- the "DeliveryType" parameter, which is filled if the scope is DELIVERY. Currently implemented possible values for this parameter are MAIL || FOOD.

- the "forUserID" parameter, which is filled if the scope is DELIVERY or VISIT. This is an optional parameter, and defines to which member family the scopes refer to. Its values are also given by reference, using "../ApplicationResources/Grammars/en-US/Users.grxml" grammar.

- the "relationship" parameter, filled optionally only if the scope is VISIT. This parameter may be used to determine the identity of the person at front door when the information gathered by other means (face recognition context source) is incomplete.

To give an example of a complex semantic meaning retrieval using the previously described grammar, in case the user at front door says "Hi there, its me Peter, I am here to see my friend Jerry.", the filled parameters are: Scope=VISIT, userID=Peter@..., forUserID=Jerry@JGAV300600.amigo.net, relationship=friend.

A very important grammar for this application is the "../ApplicationResources/Grammars/en-US/Users.grxml". Although this grammar could be hand written, we found it more convenient to use a script for automatic generation of this grammar using a template and the UMPS service. The template, "../ApplicationResources/Grammars/en-US/Users_template.grxml", contains the main structure (grammar tag, rules, meta parameters if needed) of the Users.grxml grammar, and some parts which are filled from the user profiles (see the UMPS integration section).

If the person at front door was identified by the Face Recognition Context Source, with a high enough confidence, and Scope=ENTER, the "../ApplicationResources/Grammars/en-US/4digits.grxml" grammar is used to get input from the person, specifically a random generated 4-digits code previously presented to the user by the system. The audio file recorded will be further used for person identity verification using the Speaker Recognition service.

There are also other additional grammars which used for clarification or confirmation of user answers.

The output audio data, produced by calling the corresponding methods of the VoiceIO server, is presented to the user using the loudspeakers. For each dialog act, a txt file containing the text to be transformed to audio data is provided. In some cases, if UMPS is available, personalized output is provided, using templates for the output text (i.e. "../ApplicationResources/SystemPrompts/en-US/please_say_number_template.txt").

2.8.18.4 User – System Interaction

In this section, as user, we refer to the person standing at front door and trying to interact with the application in order to enter the house, or to let a message for the house inhabitants.

Users are detected/identified by the Face Recognition context source, and the application is informed about the presence. If the user is known to the system (has been identified with a high enough confidence), the first thing the application will do is to verify the entrance rights of the user by calling UMPS.

If user is known, he is allowed to enter the house and he has a voice print, the system asks him to repeat a random generated 4 digits code. If the user spokes the correct 4-digits code, then the audio input is sent to the Speaker Recognition service to verify the identity of the user. If the identity is confirmed, then a command is sent to the door to unlock and allow the user to enter.

In all other cases, a more general interaction with the user is initiated, using EM_main.grxml, trying to determine first off all which is the scope of the presence.

If the scope is to ENTER, the application will first query CMS about inhabitants' presence in the house to inform them. If none, then the temporary-valid password are checked and if applicable the user is asked to provide its password.

If the scope is VISIT, the user will be informed on non-availability of inhabitants, and will be asked to let a message.

If the scope is EMERGENCY, and the user has a temporary-valid password, the door is unlocked, otherwise the inhabitants are informed about the event.

If the user wants to let a message for the inhabitants, then a dialog is started to get the target person (to whom the message is addressed) and the content of the message. This message will then be registered to the message board application using the IMessagingBoard::addNewMessage() web method.

2.8.18.5 Error Messages

When the EntranceManager application is started, it checks for the availability of the different services used, and an error message is shown in the console window (i.e. "UMPS service not found!", "Speaker Recognition service not found!", "Message Board service not found!").

Also, similar messages will appear in this window, if any of the requested services cannot be contacted at runtime.

If during the user-system interaction the limit number of errors is reached when trying to understand the user speech (i.e. too noisy environment), then the user is informed that the interaction cannot continue to the environment conditions, and a notification message will also appear in the application window.

2.9 Crisis Response Application (Anti-intrusion and Presence simulation)

Service Oriented Architecture (SOA) is a software architecture approach which assumes a collection of services that communicate with each other to achieve a common goal. Using the SOA principles, a service advertises its service description for potential clients. A client interested in accessing the service obtains information about the existence of a service, its applicable parameters and terms through service discovery. The Amigo project aims to achieve the widespread acceptance of home networking technology based on SOA. The Crisis Response Service (CRS) is a part of the Amigo project and specifically focuses on handling emergency situations in the home.

2.9.1 Introduction

The CRS is responsible for taking action whenever an emergency situation is detected. In this case, the CRS searches for the Emergency Response Service (ERS) based on the nature of emergency, location of the emergency and location of the ERS. The application makes use of the following Amigo Intelligent User Services (IUS): Context Aware Service Discovery (CASD), Awareness and Notification Service (ANS) and Context Management Service (CMS).

There are the following two ways to access the CRA functionality:

1. *Discover and use Emergency Response Services directly from CRS:* This approach allows the application developer interested to invoke ERS by directly invoking `DiscoverAndUseEmergencyResponseService` function of the CRS. In this case, CRS uses the CASD service to obtain a reference to the requested ERS such as an ambulance, police patrol car or fire brigade by taking into account the context information of the home (e.g. time, location) and also the context of the ERS (e.g. location, speed).
2. *Subscribe to CRS with ANS:* In this case, CRS is made aware of any crisis situation in the home by subscribing to ANS with specific rules to be aware of the emergency condition. The ANS notifies the CRS when the rule condition is satisfied. For example, CRS specifies a rule in ANS to be notified when an intruder is detected based on the ANS rules provided by the developer. On detecting the intrusion, the anti-Intrusion context source provides certain context data to ANS so that the rule is satisfied. Once the CRA obtains the notification from ANS for intrusion detection, it uses the CASD service to obtain a reference to the appropriate Crisis ERS such as a patrol car, taking into account the context information of the Intrusion (e.g. time, location) and also the context of the ERS (e.g. location, speed).

Furthermore, CRS could be used as an ***application*** or as a ***service***. In the application mode, the CRS reads information about the ANS rules and required ERS directly from the framework properties, while in the service mode, it provides the interfaces to invoke it as a service.

2.9.2 Prerequisites

The CRS service has been developed to be used as an OSGI Amigo service. Hence an OSGI Amigo framework is required to use CRS. For the comprehensive information on the OSGI Amigo, refer to the user guide located at http://amigo.gforge.inria.fr/home/components/wp3/OSGi_Framework/guides/Amigo_user_guide.pdf.

Apart from this, you need to install the following services:

Context-Aware Service Discovery (CASD) service: basemdw->discovery->trunk->Context-Aware_Service_Discovery

CASDHelper service: basemdw->discovery->trunk->CASDHelper

ANS: Refer to the ANS documentation.

ANS API: Refer to the ANS documentation.

The CRS project directory is located in GFORGE SVN at:

Basemdw -> discovery -> trunk -> CRAWithANS.

The emergency response services for the ambulance, fire brigade and police patrol could be found at:

Basemdw -> discovery -> trunk ->AmbulanceService

Basemdw -> discovery -> trunk ->FireBrigadeService

Basemdw -> discovery -> trunk ->PolicePatrolService

The sample client for CRS service could be found at the following location:

Basemdw -> discovery -> trunk ->CRS_Client

This document serves as a HOW TO guide for using CRS to discover and use an emergency response service.

2.9.3 CRS Component Description

The components collectively forming CRS are explained below. Please note that the components 1 to 6 are used for using CRS in the application mode while components 7 and 8 are used for using CRS in the service mode:

- `Activator:` A generic Amigo bundle activator
- `ANSClient` **`implements`** `ICASDServiceChanged, Lifecycle:` This is the main class which reads the necessary bundle properties to get information about the ANS rules, subscribes to ANS with the rules, initializing CASD client context source and activate the listener whenever the rule is satisfied.
- `ANSWrapper:` This class discovers the Amigo ANS service and encapsulates it. It provides the functionality of the `IManageRules` interface to an application that wants to use ANS.
- `CASDHelperClient:` Similar to `ANSWrapper`, the `CASDHelper` class discovers the Amigo CASD service and encapsulates it. It provides the functionality of the `IAmigoCASD` interface to an application that wants to use CASD. `CASDHelperClient` is a client of `CASDHelper` for using CASD functionality.
- `ANSClientNotify` **`implements`** `INotifyApplication:` This Class is originally intended for implementing the functionality to notify a user when a rule is satisfied in ANS. CRS extends the functionality of the class to also invoke the CASD service, obtain a handle to the Crisis Response Service and use the service.
- `Context Sources:` ANS requires that the context source should provide context information based on which the rule condition is satisfied. Hence an application developer needs to create an appropriate implementation of the context source for using CRS. Moreover, the CASD service requires the context information of the client in order to determine the most suitable service. Hence a client context source is also necessary.
- `CrisisResponseServiceInterface:` This interface consists of the following three methods: `DiscoverAndUseEmergencyResponseService()`, `subscribe()`, `unsubscribe()`. These methods are implemented by

`CrisisResponseServiceImpl` class. We describe these methods in detail in the User guides section.

- CrisisResponseServiceImpl: This class implements `CrisisResponseServiceInterface`. The implementation of the `DiscoverAndUseEmergencyResponseService` method uses the information provided by the developer to discover and invoke the ERS directly. The implementation of the `subscribe()` method uses the information provided by the developer to subscribe to ANS and invoking ERS whenever a rule is satisfied in ANS. The implementation of the `unsubscribe()` method is used to unsubscribe to ANS.

2.9.4 Integrated Applications

CRS uses the following WP3 and WP4 services: Context Aware Service Discovery (CASD), Awareness and Notification Service (ANS) and Context Management Service (CMS). The steps using which these services are used are shown in the Figure 42 and explained in the following:

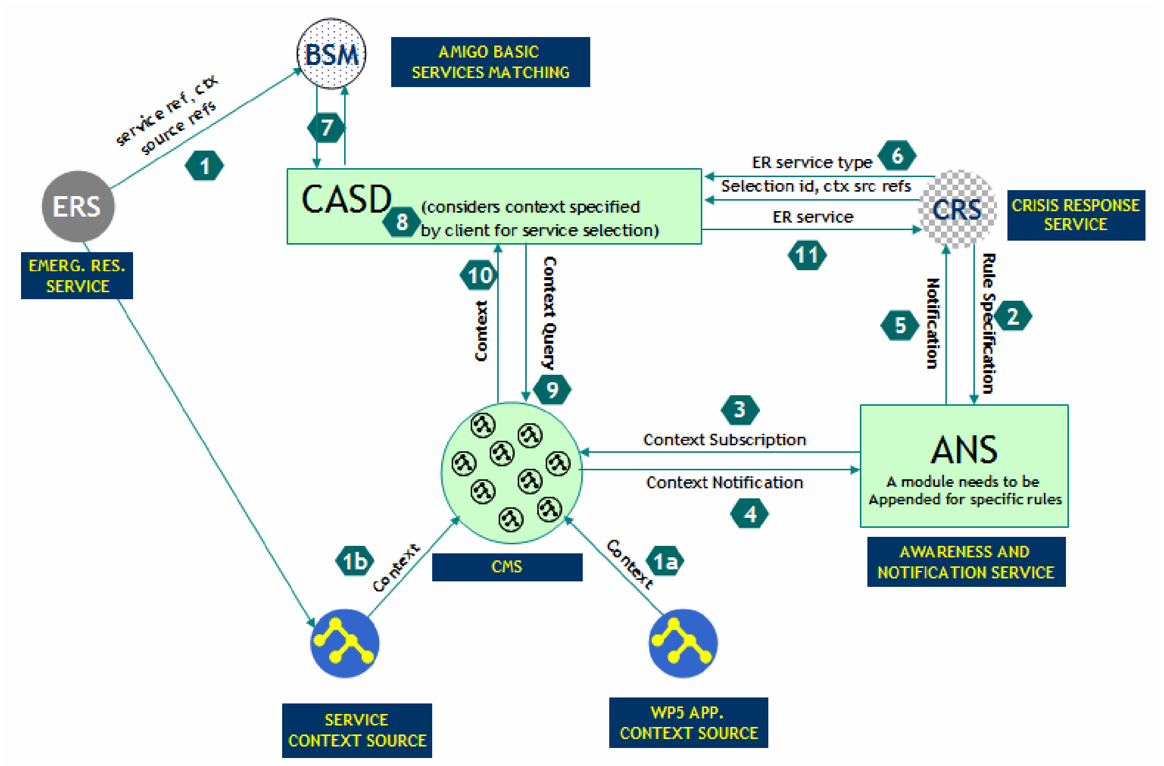


Figure 48 Interactions between Crisis Response Service and WP3 and WP4 services

1. The Emergency Response Services and the application interested to use CRS activate their context sources. (For use in the Amigo project we assume that the Emergency Response Services, though external are discoverable using Amigo Basic Service Discovery.) The context sources of the services register with the context broker so that they can be discovered. (For use in the Amigo project we assume that the context sources, though external are discoverable using Amigo CMS.)
2. The Crisis Response Service registers the rules in ANS.
3. ANS subscribes to CMS for receiving context information related to the configured rules.

4. ANS receives the context notification from CMS.
5. When the rule is satisfied, ANS notifies CRS accordingly with the alarm.
6. CRS requests the CASD service to discover the Emergency Response Service (ERS). The CRS sends the type of desired service (as requested by the developer), Service Selection Identifier (SSID) and reference to the developer context sources (home location context source).
7. On receiving the request from CRA, CASD requests Amigo Basic Service Matching (BSM) Service to return the services which match the type of service specified by the CRS.
8. CASD requests the service to provide the necessary details required to obtain the context information. The service sends its credentials, reference to its context broker and type of context.
9. CASD requests Amigo CMS to provide references to the context sources of the service and client.
10. CASD collects the context information for the services and client.
11. CASD executes the matchmaking algorithm to select the best ERS and provides its handle to the CRS.

2.9.5 User and Developer Guide

This section describes the steps involved in using CRS for the application development and extending CRS for the additional functionality.

2.9.6 User Guide: Using CRS in the Application Mode

To use CRS in the application mode, you need to configure the information about ANS rules in the `bundle.properties` file of OSCAR. Because in this mode, on activation, CRS reads the number of rules and the files containing rule descriptions from the bundle properties. For this purpose it is necessary to add the following information to the `bundle.properties` file of OSCAR.

```
#total number of rules to be used in ANS
nl.telin.amigo.cra.RULE_NUM=1
#Rule file location
nl.telin.amigo.cra.RULE_FILE1={Location of the rule file. E.g. C:/GFORGE_SVN
/basemdw/discovery/trunk/CRAWithANS/CRA_Config_Example.txt}
```

It is possible to specify multiple numbers of rules and the corresponding files in `bundle.properties`.

The format of a rule file is as follows:

```
Rule:
{XML Description of Rule to be configured in ANS}
CRS:
{Name of Emergency Response Service} {Service Selection Identifier e.g.
CLOSEST}
RuleIdentity:
{A unique string associated with the rule}
```

Note: The `RuleIdentity` should be the substring of the notification message in the rule configured in ANS.

A sample rule file is provided at the following location: `basemdw/discovery/trunk/CRAWithANS/CRA_Config_Example.txt` and also given in the following:

```
Rule:
<?xml version="1.0" encoding="UTF-8"?>
<ECARule>
<upon>
<event name="isLocatedIn" state_transition="EnterTrue">
<param>
<literal value="User.Richard.amigodomain.xyz"/>
</param>
<param>
<literal value="Close"/>
</param>
</event>
</upon>
<do>
<notification name="Notify">
<param>
<literal value="ANS_Client"/>
</param>
<param>
<literal value="Richard Etter is close to the demo site"/>
</param>
</notification>
</do>
<lifetime value="always"/>
</ECARule>

CRS:
AmbulanceService CLOSEST

RuleIdentity:
Richard
```

Please, note that in this rule file, the `RuleIdentity` is Richard and it is the first word of the message `"Richard Etter is close to the demo site"` which will be returned by ANS on the satisfaction of the rule.

2.9.7 User Guide: Using CRS in the Service Mode

To use CRS in the service mode, a sample client of the CRS service is provided in the project `CRS_Client`. It consists of the following three classes: `CRSClientActivator` (a generic amigo activator), `CRSClient` (the implementation of CRS Client) and `CRSClientContextSource` (a sample context source for the client). It should be possible to reuse most of the code for your application development. The client works as follows:

2.9.7.1 On Activation

On activation, the CRS client initiates the client context source. In most of the cases, this is the home location context source. This is done by means of the following:

```
CRSClientContextSource    clientcs    =    new    CRSClientContextSource
(manager);

clientcs.init();
```

Please, note that to represent the exact coordinates of the home location, change the values of the coordinates (longitude, latitude and altitude) in the following method of `CRSClientContextSource` class:

```
public void init()
{
      csf = manager.registerContextSource(rdfCaps);
      model = csf.getOntModel();
      // Telematica Instituut :     Latitude = 52.2328, Longitude = 6.8897
      String context = staticRDF1 +
      // Long
      "6.8897" + staticRDF2 +
      // Lat
      "52.2328" + staticRDF3 +
      // Alt
      "0.0" + staticRDF4;
}
```

Later, the `CRSClient` could use the CRS functionality by using the following methods as required:

```
            invokeCRSDirectly()
            subscribeCRS()
```

2.9.7.2 Invoke CRS Directly

The function `invokeCRSDirectly()`in the `CRSClient` class elaborates how to use CRS for discovering and using ERS directly (i.e. without using ANS). The function `DiscoverAndUseEmergencyResponseService (String CRService, String[] ctxRefs, String SSID)` provided by the CRS needs the following inputs:

2.9.7.2.1 Required Emergency Response Service

The description of the required ERS is a String. For this example client, the class CRSClient defines a sample ERS by the following:

```
      private String sampleCRS = "PolicePatrolService";
```

2.9.7.2.2 Reference to the Client Context Source

An example client context source named as `CRSClientContextSource` is provided in the `CRSClient` project.

The reference to the `CRSClientContextSource` is obtained as follows:

```
String[] refs = new String[] {clientcs.getAmigoReference(). toString()}
```

2.9.7.2.3 Specification of the Service Selection IDentifier (SSID)

To handle the emergency situations, it is desired that the ERS discovered by CRS should be the closest to deal with the crisis situation as soon as possible. Hence, the CRS Client defines a sample SSID as follows:

```
      private String sampleSSID = "CLOSEST";
```

2.9.7.2.4 Invoking the DiscoverAndUseEmergencyResponseService Function

Once the above parameters are specified, the function `DiscoverAndUseEmergencyResponseService()` responsible for discovering and invoking the ERS in the CRS is invoked as follows:

```
GenericStub stub = service.getGenericStub();

// Now test the DiscoverAndUseEmergencyResponseService Method

logger.debug("Going to invoke the DiscoverAndUseEmergencyResponseService
method");

String desc = sampleCRS;

String[] refs = new String[] { clientcs.getAmigoReference() .toString() };

String selectionIdentifier = sampleSSID;

//String ref = clientcs.getAmigoReference().toString();

String[] argNames = new String[]{"CRService", "ctxRefs", "SSID"};

Object[] argValues = new Object[]{desc, refs, selectionIdentifier};

stub.invoke("DiscoverAndUseEmergencyResponseService", argNames, argValues);
```

In this example, stub is a handle to the CRS obtained using `FindCRSService()`. The application developer needs to change only the `sampleCRS, refs` and `sampleSSID` values.

Based on this information, the CRS service will discover and invoke the CLOSEST police patrol service if any instances of the police patrol services are running.

2.9.7.3 Invoke CRS using ANS

To facilitate the use of CRS by configuring the required rules in ANS, the `subscribeCRS()` method in the `CRSClient` could be used. The method `subscribeCRS()` uses the `CRSSubscribe (String CRService, String[] ctxRefs, String SSID, String ANSRule, String ruleIdentity)` method of CRS. The first three parameters for this method are same as those required for the `DiscoverAndUseEmergencyResponseService()` method. The other two parameters `ANSRule` and `ruleIdentity` are explained in the following.

2.9.7.3.1 ANS Rule

Since it is required to configure a rule in ANS so that the CRS will be notified on satisfying the rule, the function `CRSSubscribe()` takes as input a valid ANS Rule. In the CRSClient example, a sample rule is defined as follows:

```
private String sampleRule = "<?xml version=\"1.0\" encoding=\"UTF-
8\"?>\n<ECARule>\n<upon>\n<event name=\"isLocatedIn\"
state_transition=\"EnterTrue\">\n<param>\n<literal
value=\"User.Richard.amigodomain.xyz\"/>\n</param>\n<param>\n<literal
value=\"Close\"/>\n</param>\n</event>\n</upon>\n<do>\n<notification
name=\"Notify\">\n<param>\n<literal
value=\"CRS_ANS_Client\"/>\n</param>\n<param>\n<literal value=\"Richard
Etter is close to the demo
site\"/>\n</param>\n</notification>\n</do>\n<lifetime
value=\"always\"/>\n</ECARule>\n";
```

2.9.7.3.2 ANS Rule Identity

The ANS Rule Identity is a unique string associated with the rule configured in ANS. The `RuleIdentity` should be the starting substring of the notification message in the rule configured in ANS. The CRS Client defines the following sample rule identity:

```
private String sampleRuleIdentity = "Richard";
```

Please, note that in this case, the `RuleIdentity` is Richard and it is the first word of the message `"Richard Etter is close to the demo site"` which will be returned by ANS on the satisfaction of the rule.

2.9.7.3.3 Invoking the CRSSubscribe Function

Once the above parameters are specified, the function `CRSSubscribe()` responsible for discovering and invoking the ERS in the CRS is invoked as follows:

```
GenericStub stub = service.getGenericStub();

logger.debug("Going to invoke the Subscribe method");

String desc = sampleCRS;

String[] refs = new String[] { clientcs.getAmigoReference().toString()
};

String selectionIdentifier = sampleSSID;

String rule = sampleRule;

String ruleIdentity = sampleRuleIdentity;

String SSID = sampleSSID;

logger.debug("SampleRule = \n" + sampleRule);

//public int subscribe(String CRService, String[] ctxRefs, String
SSID, String ANSRule, String ruleIdentity);

String[] argNames = new String[]{"CRService", "ctxRefs", "SSID",
"ANSRule", "ruleIdentity"};

Object[] argValues = new Object[]{desc, refs, selectionIdentifier,
rule, ruleIdentity};

Object Response = stub.invoke("CRSSubscribe", argNames, argValues);

sampleReturnedRuleID = Integer.parseInt((String) Response);
```

In this example, stub is a handle to the CRS obtained using `FindCRSService()`. The application developer needs to change only the `sampleCRS, refs, sampleSSID, rule` and `ruleIdentity` values.

Based on this information, the CRS service will configure the given rule in ANS and after receiving notification on satisfying the rule, will discover and invoke the CLOSEST police patrol service if any instances of the police patrol services are running.

2.9.7.3.4 Unsubscribing to the CRS

The invocation of `CRSSubscribe` returns a `RuleID`, which should be used to unsubscribe the rule from ANS. The `unsubscribeCRS()` method in the CRSClient could be used for this purpose and it works as follows:

```
GenericStub stub = service.getGenericStub();

logger.debug("Going to invoke the unsubscribe method");
```

```
      //public void unsubscribe(int ANSRuleID);
      if (sampleReturnedRuleID != -1) {
            String[] argNames = new String[]{"ANSRuleID"};

            Object[]          argValues          =          new
      Object[]{Integer.valueOf(sampleReturnedRuleID)};
            stub.invoke("CRSUnsubscribe", argNames, argValues);
            sampleReturnedRuleID = -1;
      } else logger.warn("Looks like you don't have a valid Rule ID returend
      from ANS. Check whether you have configured rule correctly.");
```

In this example, stub is a handle to the CRS obtained using `FindCRSService()`. The application developer needs to change only the value of `sampleReturnedRuleID`. In principle, this value is automatically assigned when the function `CRSSubscribe()` returns.

2.9.8 Developer Guide: Extending CRS for Additional ERSes

In the ***application*** mode, since we have chosen the approach of reading rules and the respective CRSes using the locations specified in the property file and as much as number of rules could be specified, it is not tedious to extend CRA for the desired functionality. A developer will need to change a part of `ANSClient` and `ANSClientNotify` classes, create the context sources and specify rules for ANS.

We have developed three types of ERS, namely `AmbulanceService`, `FireBrigadeService` and `PolicePatrolService`. For the additional ERSes, the following function in the `ANSClientNotify` class needs to be extended to invoke additional ERSes obtained from the CASD service:

```
private void invokeCRService(String CRService, GenericStub stub)
```

One example of using `AmbulanceService` is given in the code and it is as follows:

```
if (CRService.equals ("AmbulanceService")) {
      String[] argNames = new String[]{"location"};
      Object[] argValues = new Object[]{"Telematica Insituut"};
      Object response = null;
      try {
            response = stub.invoke("orderAmbulance", argNames, argValues);
      } catch (Exception e) {
            logger.debug(e.toString());
      }
      logger.debug("Result of the Ambulance Service query is:" + response);
}
```

The CASD service requires the context information of the client in order to determine the most suitable service. Since CRS is aimed at use within home, the client location is the same as the home location. Hence the CRS project in GFORGE also contains a sample `HomeLocationContextSource`. This context source could be modified for the customized client context. A context source reference is passed to `ANSClientNotify` to further invoke CASD service. The relevant code in `ANSClient` is as follows:

```
      clientcs = new HomeLocationContextSource (manager);
      clientcs.init();
      String[] refs = new String[] { clientcs.getAmigoReference().toString()
      };
      ANSClientNotify service = new ANSClientNotify (refs, ruleIdentity,
      CRServices, SSIDs, casdHelper);
```

For more information on how to write rules for ANS and developing context sources so that the rules specified in ANS will work, please refer to ANS documentation.

In the ***service*** mode, we have provided a sample client in the `CRSClient` project. With some minor changes, this client could be used as it is for accessing CRS. For the additional ERSes, the following function in the `CrisisResponseServiceImpl` class needs to be extended to invoke additional ERSes obtained from the CASD service.

2.9.9 Conclusions

This document serves as a HOW TO guide for using CRS to invoke an emergency response service. To be able to use CRS in the ***application*** mode, a developer needs to set Amigo bundle properties for the number of rules and the file locations containing their descriptions. Furthermore, it is also necessary to write in the rule file `relevant rules`, the required `Crisis Response Service`, `Service Selection Identifier` and `Rule Identity`. Furthermore, if not implemented before, a context source needs to be developed and appropriate changes need to be done to the `ANSClient` class and `ANSClientNotify` class.

In the ***service*** mode, we have provided a sample client in the `CRSClient` project. With some minor changes, this client could be used as it is for accessing CRS. In this case, the application developer needs to change only the `sampleCRS, refs, sampleSSID, rule` and `ruleIdentity` values.

3 Conclusions

The main aim of WP5 has been to test and verify the suitability of Amigo middleware an Amigo intelligent users services to design and develop Home care and safety applications. The process for selecting which applications were been developed started within WP1 by asking end users for what type of functionalities and applications they would like to have at their home in the near future. The result of it was the following:

- Comfortable environment. Automation and control of:
  - Heating, air conditioning
  - Household appliances
  - A quite number of sensors
  - Windows, blinds, doors,
  - other
- Safety and Security. Detection and following actions
  - Water and gas leakage
  - Fire, smoke
  - Buglers, anti-intrusion
- Less housework and easiness
- Low resources consumption
  - (electricity, water, gas, detergents,…)
- Wellness and Health
  - Feeling safety, protected
  - Children, elderly
  - Health measurements
  - Eating healthy, special diets
  - Being monitored
  - Being advised and guided
  - Doing exercise

Combining the results of WP1 and the competences of industrial partners within WP5, the applications selected to be developed are included in the following sub domains within the domain of Home care and Safety:

- *Comfortable environment*
  - Appliances manager
  - Allow the entrance & Get visitor's message
  - Comfort system
- *Safety and Security*
  - Technical alarms
  - Crisis Response Application

- *Wellness and health*
    - Health Manager
    - Daily Life Cycle monitor
    - Food Management

The development of previous selected applications has been supported on most of the parts of Amigo middleware developed within WP3 and most of intelligent users services developed within WP4. There are few basic services that are not been used in WP5 applications and they are, for example, accounting and billing and security services. The rest of the results coming from WP3 and WP4 have been extensively used and tested.

Regarding to the functionalities and benefits for the users provided by WP5 applications, the user evaluation and opinion obtained after some user's tests has been very optimistic and positive.